2007-08-29

Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science

Sarah Jane Delany
Technological University Dublin, sarahjane.delany@tudublin.ie

Michael Madden
NUI Galway

Recommended Citation

Delany, S.J. & M. Madden, (2007). *Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science*, Dublin 29-31 August. doi:10.21427/wmr8-km59

Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science

Sarah Jane Delany and Michael Madden (eds)

S. J. Delany and M. Madden (Eds.)

AICS 2007

Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science

29th to 31st August 2007

School of Computing,
Dublin Institute of Technology,
Kevin Street,
Dublin 8,
Ireland.

Printed by:

Cahill Printers Ltd,
IDA Business & Technology Park,
Clonshaugh,
Dublin 17

Preface

These proceedings contain the papers that were accepted for publication at AICS-2007, the 18th Annual Conference on Artificial Intelligence and Cognitive Science, which was held in the Dublin Institute of Technology; Dublin, Ireland; on the 29th to the 31st August 2007. AICS is the annual conference of the Artificial Intelligence Association of Ireland (AIAI). Since its inception in 1988, it has provided a forum for the exchange of ideas and the presentation of results relating to work conducted in Ireland and worldwide.

The conference spans both basic and applied research in the areas of Artificial Intelligence and Cognitive Science, broadly construed. This year's papers covered topics including constraint satisfaction, artificial life, intelligent user interfaces, information retrieval, machine learning, and multi-agent systems. All submissions were reviewed by three anonymous reviewers; 45% were accepted for oral presentation, 24% were accepted for poster presentation and the remaining 31% were not accepted.

This year's conference also featured thought-provoking and interesting keynote talks by two distinguished AI researchers. Dr Mehran Sahami, a senior research scientist from Google, spoke about trends in making use of large Web datasets and the potential they offer for further advancing research and development in Artificial Intelligence. Dr Eamonn Keogh, a professor in the University of California, Riverside, spoke about the lack of reproducibility of results in data mining and machine learning. He argued that this is crippling research progress and allowing a large number of false research findings go unchallenged and enter the popular consciousness as true.

We thank all authors for submitting their work to AICS, and the PC members and additional reviewers who all provided thoughtful analyses and critiques of the submissions. We are also grateful for the help and support we received from DIT's School of Computing, in particular the administrative support from Denise Murray and technical support from Michael Gleeson. We also appreciate the advice received from Derek Bridge, Chair of AIAI, and other members of AIAI who have organised this conference previously.

We sincerely thank this year's sponsors of AICS: Dublin Institute of Technology, School of Computing; Enterprise Ireland, ARC Informatics Commercialisation Team; and Google. Their financial support has enabled the participation of our keynote speakers, the provision of a substantial prize for best paper, and a convivial social programme, while ensuring that participation remained affordable for attendees.

Finally, we are grateful to all of the researchers who support AICS each year by their attendance at the conference, and we hope that the papers in these proceedings will be a useful resource to AI and Cognitive Science researchers internationally.

Michael Madden & Sarah Jane Delany
AICS-2007 Programme Chairs
August 2007.

Organising Committee

Conference Chairs

Sarah Jane Delany	Dublin Institute of Technology
Michael Madden	NUI Galway

Programme Committee

David Bell	Queen's University, Belfast
Michaela Black	University of Ulster, Coleraine
Derek Bridge	University College Cork
Ken Brown	University College Cork
Arthur Cater	University College Dublin
Fintan Costello	University College Dublin
Brian Crean	Galway-Mayo Institute of Technology
Norman Creaney	University of Ulster, Coleraine
Fred Cummins	University College Dublin
Pádraig Cunningham	University College Dublin
John Dunnion	University College Dublin
Malachy Eaton	University of Limerick
Tim Fernando	Trinity College Dublin
Josephine Griffith	NUI Galway
Ray Hickey	University of Ulster, Coleraine
Mark Humphrys	Dublin City University
John Kelleher	Dublin Institute of Technology
Weiru Liu	Queen's University Belfast
Brian Mac Namee,	Dublin Institute of Technology
Lorraine McGinty	University College Dublin
Paul Mc Kevitt	University of Ulster, Magee
David McSherry	University of Ulster, Coleraine
Diarmuid O'Donoghue	NUI Maynooth
Colm O'Riordan	NUI Galway
Barry O'Sullivan	University College Cork
Geároid O'Neill	University of Limerick
Ronan Reilly	NUI Maynooth
Stephen Sheridan	Blanchardstown Institute of Technology
Barry Smyth	University College Dublin
Humphrey Sorenson	University College Cork
Ray Walshe	Dublin City University

Additional Reviewers

Dara Curran	University College Cork
Conor Nugent	University College Cork

Acknowledgements

The programme committee of AICS07 would like to gratefully acknowledge the sponsors of this year's conference:

School of Computing, Dublin Institute of Technology;

ARC Informatics Commercialisation Team, Enterprise Ireland;

Google.

Table of Contents

Oral Presentations

Poster Presentations

An Evaluation of One-Class Classification Techniques for Speaker Verification

Anthony Brew, Marco Grimaldi, and Pádraig Cunningham

School of Computer Science and Informatics, University College Dublin
{Anthony.Brew,Marco.Grimaldi,Padraig.Cunningham}@ucd.ie

Abstract. Speaker verification is a challenging problem in speaker recognition where the objective is to determine whether a segment of speech in fact comes from a specific individual. In supervised machine learning terms this is a challenging problem as, while examples belonging to the target class are easy to gather, the set of counter-examples is completely open. In this paper we cast this as a one-class classification problem and evaluate a variety of state-of-the-art one-class classification techniques on a benchmark speech recognition dataset. We show that of the one-class classification techniques, Gaussian Mixture Models shows the best performance on this task.

1 Introduction

In speaker recognition research two separate problem categories are identified; speaker identification and speaker verification [1]. In machine learning, speaker identification is an n-class supervised learning problem where the query sample is matched to one of n classes in the training data. Speaker verification might be considered a binary classification problem in that the objective is to determine whether or not the query is from the individual whose identity is claimed for the query. Given that binary classification is normally easier than multi-class classification, speaker verification would appear to be an easier problem to solve than speaker identification. However, real-world examples of the speaker verification problem, as arising for instance in security applications, are very challenging because of their *open* nature. If the utterances of an individual are the examples of the class to be recognised then the *non-class* examples cover everything else. For this reason it is worth analysing the merit of casting this as a one-class classification problem rather than a binary classification problem.

One-class classifiers (OCCs) have emerged as a set of techniques for situations where labeled data exists for only one of the classes in a two-class problem. For instance, in industrial inspection tasks, abundant data may only exist describing the process operating correctly. Training data describing the myriad of ways the system might operate incorrectly are difficult or impossible to gather. The philosophy behind the OCC approach is to develop a classifier that *characterises* the target class, and thus can distinguish it from all counter-examples.

A related problem arises where negative examples exist, but their distribution cannot be characterised. For example, it is easy to provide characteristic

examples of the writings of Shakespeare but impossible to provide examples of the counter-class (material *not* by Shakespeare). While such problems are also appropriate for the OCC approach, the motivation is slightly different – counter examples are in fact available but it is difficult to construct a set of counter examples with good coverage of the universe of possible counter examples.

In speaker verification the problem is generally cast as a binary problem, unfortunately the imposter is impossible to accurately model. Although we recognise the fact that use of an impostor model will dramatically improve performance, we believe the area merits a base evaluation, where OCCs are trained solely on target data so that direct comparisons can be fairly made. The evaluation presented here is carried out on the CHAINS corpus introduced by Cummins et al. [2].

The paper proceeds with an overview of the speaker recognition research area in section 2 and a brief review of the relevant OCC techniques in section 3. The main results of the evaluation are presented in section 4 and the paper concludes with a summary and some suggestions for future work in section 5.

2 Speaker Verification

Speaker recognition systems aim to extract, characterize and recognize the information enclosed in the speech signal conveying the identity of a speaker. The general area of speaker recognition includes two fundamental tasks: *speaker identification* and *speaker verification* [3, 4]. Speaker identification is the task of assigning an unknown voice to one of the speakers known by the system: it is assumed that the voice must come from a fixed set of speakers. Thus, the system must solve a n-class classification problem and the task is often referred to as *closed-set* identification.

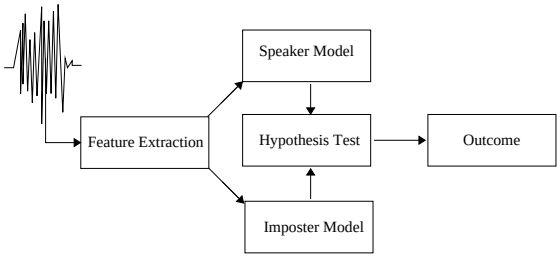


Fig. 1. Speaker Verification System.

On the other hand, speaker verification refers to the case of *open-set* identification: it is generally assumed that the unknown voice may come from an impostor, not all the speakers accessing the system are known. In this case, the standard approach is based on a likelihood ratio test to distinguish between the two hypotheses: the test speech comes from the claimed speaker or from an impostor. Furthermore, depending on the specific application, speaker verification systems can work in a text-dependent or text-independent setup. In text-dependent applications the verification system has prior knowledge of the text to be spoken (e.g. a pass-phrase). In a text-independent application, no prior knowledge of the text to be spoken is provided to the system.

Generally, speaker verification systems are composed of three main components as shown in Figure 1: a front-end responsible for signal processing and feature extraction, a model for each speaker allowed to access the system and a model for impostor detection. The big advantage of the OCC approach to speaker verification is that there is no need for an impostor model as the classifier is trained on the speaker model only. In the next two sections we introduce each individual component of the verification system.

2.1 Front-end Processing and Feature Extraction

As a first step, the front-end module of the verification system generally performs speech activity detection to remove the non-speech portion of the signal. Next, features embodying information on the speaker identity are extracted from the speech signal. Finally, the front-end implements some form of channel compensation in order to remove those spectral characteristics that are dependent on the acquisition channel (e.g microphone) and do not reflect the speaker identity.

In most speaker verification and identification systems, some form of spectral-based parametrization is used to encode the speech in machine readable form. Typically short-term analysis (about 20 ms) is used to compute a sequence of magnitude spectra. Most commonly, the spectra obtained are then converted into cepstral coefficients and the frequency scale warped into the Mel scale [3].

In this work, conventional Mel Frequency Cepstral Coefficients (MFCCs) feature vectors are employed for speech parametrization. 25 MFCCs are used for speech parametrization, extracted using a Hamming window of about 20 ms. The zeroth cepstral coefficients (the DC level of the log-spectral energies) are not used in the feature vector.

2.2 Speaker Modeling

The feature vectors extracted from the training speech material are used to create a set of speaker models, to verify if the test speech sample belongs to one of the speakers in the pool. The modeling of a speaker may be implemented according to various approaches, i.e. k nearest neighbour (k-NN), neural networks, hidden Markov models (HMMs) and support vector machines (SVMs). Generally, the selection of the model adopted is largely dependent on the type of speech used, the expected performance and the computational and storage cost [4]. From published results (e.g. [1]), HMM-based systems generally produce the best performance and in the case of text-independent applications single state HMMs – also known as Gaussian mixture models (GMMs) – are the most commonly used. Neural networks have been largely tested in this context also, however some of their shortcomings (such as the fact that the optimal structure has to be selected by trial-and-error procedures) have been judged crucial in the area of speaker verification [3, 4]. SVMs, on the other hand, have been the subject of recent studies [3] aimed at adapting this extremely powerful technique to the problem of speaker verification.

2.3 Impostor Modeling

Two main approaches are used to obtain the impostor model used in the likelihood ratio test implemented in speaker verification systems. The first approach – known as likelihood sets, cohort or background speakers [3, 4] – uses a set of other speakers to cover the space of alternative hypotheses. The impostor score is usually computed as a function (e.g. max, average) of the match scores obtained from the alternative models. It is generally recognized that this approach requires a speaker-specific background speaker set to obtain the best performance [3]. The second approach to impostor modeling uses a single speaker-independent model trained on speech from a large number of speakers. This approach is usually referred to as general model, world model or universal background model (UBM) [3]. The main advantage of the UBM approach is that a single speaker-independent model is trained and then used for all the speakers in the pool. This approach has become the predominant approach used in speaker verification systems. Generally, these two approaches can be applied to any speaker modeling technique [4].

3 One-Class Classification (OCC)

When employing binary classification we attempt to train a known speaker against anything that is 'not' from the speaker. This is an unfortunate scenario, as to sample everything that is 'not' is an impossible task. We are training with a class that is statistically well-sampled versus a class that is not. This statistical imbalance in the training set may lead to the creation of a system that does not generalise well when run against non-training data.

The area of OCC is well adapted to such problems; one builds a model that creates a boundary around the well-sampled target distribution that rejects all but a small percentage f of target examples and hence hopes to be able to identify $(100 - f)\%$ of the target while rejecting as many of the outlier class as possible. Most OCCs will produce a score for a given example and if it lies above a given threshold it is classified as a member of the target class.

The data we have is a sequence of sets of Mel cepstral coefficients where each set represents a time slice of 20ms of speech and there is an overlap between time slices of 10ms. Each individual slice is not particularly informative to predict the class of the data. It is assumed that groups of slices are taken from the same single source speaker. An approach that is used in the n-class problem is to take many slices accounting for a given period of speech and aggregate the scores to strengthen the evidence that this window over a sequence of slices comes from a given source [1]. Here we propose a similar strategy, which is explained in section 3.2.

3.1 Selection of Classifiers Used

In this work we chose four OCCs to compare: a single Gaussian, a GMM, a k-NN based approach and an approach based on SVMs, the support vector domain description (SVDD) [5].

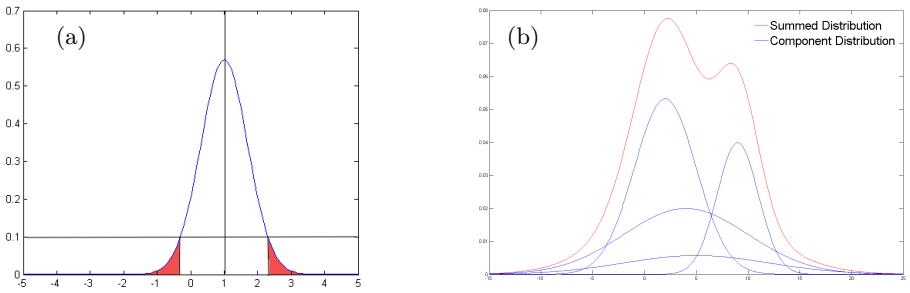


Fig. 2. In (a) a single Gaussian is used to model the underlying target data, the value of the gaussian function is thresholded so that when a value of less than the threshold is found the item will be rejected. Whereas (b) (a GMM) shows by using many individual Gaussian models and weighting them, more complex distribution shapes can be formed.

Single Gaussian: A simple model for any problem is to assume the data is drawn from the Gaussian distribution [6]. This model assumes that the data fits a unimodal convex data description. The function that determines the score, where μ is the mean of all the 'target' points, Σ it the covariance matrix of the target points and d is the dimensionality of the problem given by the classifier is :

$$p(\mathbf{x}, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|} exp\left(\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{z}-\mu)\right)$$

Gaussian Mixture Models (GMMs): GMMs can model more complex underlying distributions. As the name suggests GMMS are the combination of several Gaussian models (a weighted sum). The underlying Gaussians of the GMM have been shown to represent the characteristic spectral shapes of the phonetic sounds that make up a person's voice [1]. In these experiments we use only diagonal covariance matrices for each Gaussian. It is argued in [1] that this limits the computational complexity of the problem and adding more underlying mixtures of diagonal covariance, is equivalent to modeling with fewer full covariance matrices. The function that determines the score, where p is the Gaussian of an individual model, the α_i's are mixture weights which sum to one, and k is the number of individual Gaussian models used is given by:

$$p_{GMM}(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i p(\mathbf{x}, \mu_\mathbf{i}, \Sigma_i)$$

Support Vector Domain Description (SVDD): The SVDD finds a sphere of minimum radius that encloses all of the target data. This is cast as a minimisation problem where one finds a radius $\mathbf{R}$ and centre $\mathbf{c}$ such that the following

minimisation problem is solved. Data that lies further than a given distance from the centre of the sphere is labeled as an outlier.

$$\min \mathbf{R}^2 \;\; s.t. \\ \langle \mathbf{x}_i - \mathbf{c}, \mathbf{x}_i - \mathbf{c} \rangle \leq \mathbf{R}^2 \;\; \forall \;\; \mathbf{x_i} \in target$$

By replacing the inner product in the above problem with a 'kernel' function, more flexible decision boundaries may be found. Once the optimisation problem is solved, new points that are further than a given distance from the centre are labeled as outliers. Details of the mathematical derivation and details of the method can be found in [5]. The kernel function that is used in this work is the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2}$$

The σ parameter is selected for the kernel depending on the classification task at hand. It is often known as the width parameter and controls the flexibility of the decision boundary. If σ is set too high the model will tend to under-fit the data and if it is set too small it will over-fit the data.

***k*-Nearest Neighbour:** We carried out a comparison of several nearest neighbour techniques to identify the variation producing the best results on a single time slice, as determined by the highest area under the ROC curve (AUC)[7]. The k-NN method we chose takes the magnitude of the average directional vector to the k nearest neighbours as its output to threshold against.

3.2 Aggregation of Scores

Each of the above classifiers produces a score, if the score is above a given threshold it is considered to be from the 'target' class and otherwise it is considered to be an 'outlier'. Since each individual score is a high-variance prediction of the class, it makes sense to aggregate a sequence of scores when making a prediction.

The component scores from classifiers do not directly represent the probability of the item belonging to a class so in order to combine scores it is better to convert the raw scores to probabilities. To achieve this, the score for the item belonging to the target and the score for it belonging to the outliers were normalised to sum to one.

To combine the probabilities for individual classification we used a simple summation, a strategy that has been shown to give good results [8]. Although the alternative product rule follows directly from a Baysian viewpoint, under the assumption of independence, it can dramatically amplify errors as more slices are added [9]. The sum rule is much less sensitive to estimation error at the single slice level [9] and hence was used for the evaluation presented here.

4 Evaluation

The objective in the evaluation is to assess the performance of OCC techniques on the problem of speaker verification. As explained in section 2 the OCC will have the advantage that the problem of designing and constructing an impostor model is avoided.

Thus we are interested in the absolute performance of OCCs trained on 'target' data only. And we are interested in a comparative analysis of the OCCs described in section 3 against each other. We expect that GMM's will outperform other classifiers at a cepstral level as it has been shown that the underlying Gaussian components of the model inherently model the underlying distributions of the phonetic sound production [10].

4.1 Experimental Setup

The CHAINS corpus [2] is the result of an effort to provide a speech database expressly designed to characterize speakers as individuals[1]. The corpus contains the recordings of 36 speakers obtained in two different sessions with a time separation of about two months taken on two different recording environments. Across the two sessions, each speaker provides recordings in six different speaking styles.

In the experiments conducted here we used one speaking style SOLO, where 16 speakers read a prepared text alone. We only used speech from one recording session so that we would not have to manage problems imposed by different channel effects between different microphones. The training set was made up of speakers reading ten sentences making up on average 24 seconds on speech per speaker. The test set was made up of speakers reading nine sentences later in the same session making up on average 16 seconds one speech per speaker. As noted above, we train only using target data to provide a fair comparison between classifiers.

In order to select the parameters of the base classifiers we built them on the individual slice level first. The parameters for each classifier (number of mixture models for GMM, σ for the SVDD, etc.) were selected by using a consistency criterion. This criterion selects to reject $f\%$ (in our case 10%) of individual target slices and then uses more and more complex parameters for the classifier until the model becomes statistically unstable. This process defines the most complex classifier, which can still reliably be trained on the data [11]. For k-NN the simplest model was already unstable, so to select k we ran cross validation and found a best value to use ($k = 40$). It should be noted that parameter selection was done for each individual speaker rather than on a global basis.

On the training set the summed scores of the target class were thresholded to reject 10% of the target class in order to provide a tight decision boundary around the target class.

[1] The corpus is freely available for research purposes from http://chains.ucd.ie.

4.2 Discussion of Results

When an evaluation on a single 20 ms time slice was done on the data we found a high variability of scores between speakers (Fig3a.). With more slices all classification strategies increased performance and this trend continued even at the 10 second mark (Fig3b.). A very wide range of performances was found across all speakers (Fig4.). This is highlighted by the fact that, although on average the SVDD was outperformed by all other classifiers, it was the classifier which achieved the best performance on an individual speaker with a false positive rate over a 10 second window of 11.54%.

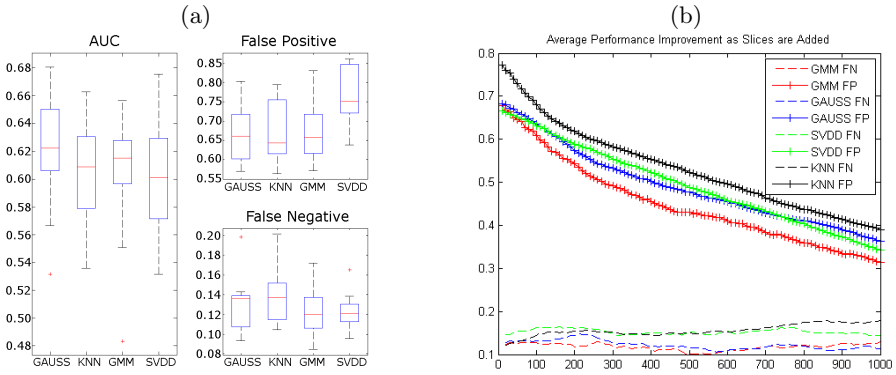


Fig. 3. In (a) it is clear that, working with a single slice only, classifiers have difficultly deciding whether the object comes from the 'target' or 'outlier' class. Over the 16 speakers the best false positive rate found was 0.56 while false positive error rates as high as 0.86 can be observed for the SVDD. In (b) the average performance increases as the number of slices is increased, this is seen to be continuing to rise even when 1000 slices ($\sim$ 10 seconds) of speech is used in the classification

By looking at the false negative scores it can be seen that on average none of the classifiers hit their trained target of 10% rejection during test and rejected more than this base amount on average.The GMM and Gaussian model best fitted their target distribution between training and test sets. As mentioned in [10] the average speech spectrum contains speaker specific information and for this reason was not removed in these experiments. The average speech spectrum can vary considerably over even short periods of time [10] and so this shift may account for the drift between the trained false negative rate differing from the values attained on the training set.

5 Conclusions and Future Work

The objective with this work was to assess whether state-of-the-art one-class classification techniques are effective for speaker verification. The evaluation presented here shows that GMM affords the best performance on average of the four

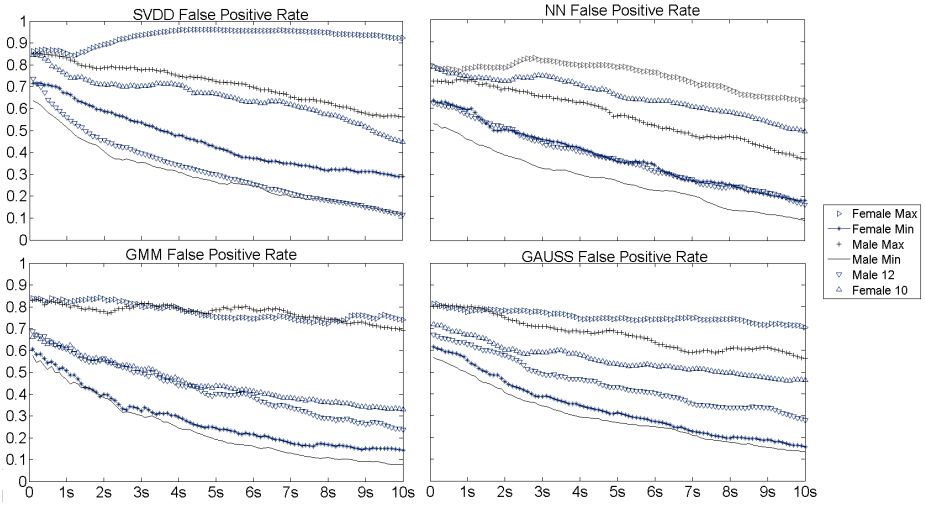


Fig. 4. It can be seen that for some individuals OCC techniques yielded good results when compared against other speakers. It is also noted that certain classification strategies preformed better for some speakers than for others indicating that model selection may also need to be considered when building an OCC for a given speaker.

classification techniques examined. As expected, the accuracy increases as the speech window used for classification is increase (i.e. more slices are added). While the accuracy on 10 seconds of speech is adequate for some speakers (i.e. $< 10\%$ FP), it is not adequate for all speakers.

It has been seen that different classification models perform better for different speakers and it would be interesting to try to tease out the reasons for these differences. Some speakers failed to perform well across all classification strategies (e.g. 'Female Max' Fig4.). These poor performing speakers may merely sit very close to one another in a region of feature space and so the spread of their underlying cepstral distributions overlaps more prominently than with other speakers. This could be investigated by looking at cross correlation matrices to see which speaker the false positives for a given individual comes from and in what percentage.

It has been suggested that the future direction of speaker verification will be in the use of higher level speech features [12] that capture not only the individual time slices but also the temporal information. While the GMM-UBM model is the best approach for a slice by slice level, active research is looking at the combination of sequence kernels for SVMs. The combination of these classifiers with the GMM-UBM has shown considerable promise [13]. An investigation into an extended feature space would seem appropriate.

The next step in this evaluation is to compare this against a binary classification approach where a broad set of speakers is sampled to produce representative training examples of the non class. We will also employ limited outlier data for

thresholding in the OCCs – this can be expected to improve on the performance of the OCCs presented here.

References

1. Reynolds, D.: Speaker identification and verification using Gaussian mixture speaker models. Speech Communication **17** (1995) 91–108
2. Cummins, F., Grimaldi, M., Leonard, T., Simko, J.: The CHAINS corpus: CHAracterizing INdividual Speakers. In: Proc of SPECOM'06. (2006) 431–435
3. Bimbot, F., Bonastre, J., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-Garcia, J., Petrovska-Delacretaz, D., Reynolds, D.: A tutorial on text-independent speaker verification. EURASIP Journal on Applied Signal Processing **4** (2004) 430–451
4. Reynolds, D.: An overview of automatic speaker recognition technology. In: Proc. Int. Conference on Acoustics, Speech, and Signal Processing. (2002)
5. Tax, D.M.J., Duin, R.P.W.: Support vector domain description. Pattern Recogn. Lett. **20** (1999) 1191–1199
6. Tax, D.M.J.: One-class classification. PhD thesis, Delft University of Technology (2001)
7. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30** (1997) 1145–1159
8. Taniguchi, M., Tresp, V.: Averaging regularized estimators. Neural Computation **9** (1997) 1163–1178
9. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on **20** (1998) 226–239
10. Reynolds, D.A., Rose, R.C.: Robust text-independent speaker identification using gaussian mixture speaker models. Speech and Audio Processing, IEEE Trans. **3** (1995) 72–83
11. Tax, D.M.J., Muller, K.R.: A consistency-based model selection for one-class classification. In: ICPR. Volume 3. (2004) 363–366
12. Reynolds, D.A.: Channel robust speaker verification via feature mapping. In: ICASSP. (2003) II–53–6 vol.2
13. Wan, V., Renals, S.: Speaker verification using sequence discriminant support vector machines. Speech and Audio Processing, IEEE Trans **13** (2005) 203–210

Using Relaxations to Improve Search in Distributed Constraint Optimisation *

David A. Burke and Kenneth N. Brown

Centre for Telecommunications Value-Chain Research
and Cork Constraint Computation Centre
Dept. Computer Science, University College Cork, Ireland

Abstract. Densely connected Distributed Constraint Optimisation Problems (DisCOP) can be difficult to solve optimally. Finding good lower bounds on constraint costs can help to speed up search, and we show how lower bounds can be found by solving relaxed problems obtained by removing inter-agent constraints. We present modifications to the ADOPT DisCOP algorithm that allow an arbitrary number of relaxations to be performed prior to solving the original problem. We identify useful relaxations based on the solving structure used by ADOPT, and demonstrate that when these relaxations are incorporated as part of the search it can lead to significant performance improvements. In particular, where agents have significant local constraint costs, we achieve over an order of magnitude reduction in the messages exchanged between agents.

1 Introduction

Many combinatorial decision problems are naturally distributed over a set of agents: e.g. coordinating activities in a sensor network [1], or scheduling meetings among a number of participants [2]. Distributed Constraint Reasoning (DCR) considers algorithms explicitly designed to handle such problems, searching for globally acceptable solutions while balancing communication load with processing time [3]. Many algorithms have been proposed that consider both satisfaction (DisCSP) and optimisation (DisCOP), including ADOPT [4]. However, ADOPT's efficiency decreases as the size and density of the network of agents increase [4, 5]. Search in ADOPT can be reduced if good lower bounds on costs are available. In this paper, we show how to generate effective lower bounds through problem relaxation. Relaxation changes a problem such that an optimal solution to the relaxed problem is a lower bound on the optimal solution to the original problem.

Relaxations have previously been applied to DisCSP, where they have been used to find solutions for over-constrained satisfaction problems [6]. In this paper, we investigate relaxation for DisCOP by removing selected inter-agent constraints. We present a relaxation framework, ADOPTRELAX, that allows ADOPT to be run in multiple phases, allowing one or more relaxed versions of the problem to be used when solving the original problem. Lower bound information gathered

* This work is supported by Science Foundation Ireland under Grant No. 03/CE3/I405

by the agents in one phase is used as input to the next, allowing portions of the search space to be pruned. While the concept of computing and re-using lower bounds dynamically during search has been explored in centralised constraint optimisation [7], this is the first investigation of such methods for DisCOP. The idea of using multiple levels of relaxations was first introduced in [8], but this also has not been investigated in a distributed environment. We identify graph-based relaxations that are of particular use with the search structures used by ADOPT, and we show that incorporating these relaxations can significantly improve performance as the size and density of the network of agents increases. In particular, where agents have significant local constraint costs we show over an order of magnitude reduction in messages passed.

2 Distributed Constraint Optimisation and ADOPT

A Distributed Constraint Optimisation Problem consists of a set of *agents*, $A = \{a_1, a_2, ..., a_n\}$, and for each agent a_i, a set $X_i = \{x_{i1}, x_{i2}, \ldots, x_{im_i}\}$ of *variables* it controls, such that $\forall i \neq j \ X_i \cap X_j = \emptyset$. Each variable x_{ij} has a corresponding domain D_{ij} of values that it may be assigned. $X = \bigcup X_i$ is the set of all variables in the problem. $C = \{c_1, c_2, \ldots, c_t\}$ is a set of *constraints*, where each c_k acts on a subset of the variables $s(c_k) \subseteq X$, and associates a cost with each tuple of assignments to these variables $c_k : \prod_{ij:x_{ij} \in s(c_k)} D_{ij} \rightarrow \mathbb{N} \cup \{\infty\}$, where a cost of infinity indicates a forbidden tuple. The *agent scope*, $a(c_k)$, of c_k is the set of agents that c_k acts upon[1]: $a(c_k) = \{a_i : X_i \cap s(c_k) \neq \emptyset\}$. An agent a_i is a *neighbour* of an agent a_j if $\exists c_k : a_i, a_j \in a(c_k)$. A *global assignment*, g, is the selection of one value for each variable in the problem: $g \in \prod_{ij} D_{ij}$. A *local assignment*, l_i, to an agent a_i, is an element of $\prod_j D_{ij}$. For any assignment t and set of variables Y, let $t_{\downarrow Y}$ be the projection of t over the variables in Y. The global objective function, F, assigns a cost to each global assignment: $F : \prod_{ij} D_{ij} \rightarrow \mathbb{N} :: g \mapsto \sum_k c_k(g_{\downarrow s(c_k)})$. An optimal solution is one which minimises F. The solution process, however, is restricted: each agent is responsible for the assignment of its own variables, and thus agents must communicate with each other, describing assignments and costs, in order to find a globally optimal solution.

ADOPT [4] is a complete DisCOP algorithm where agents execute asynchronously. Initially, the agents are prioritised into a Depth-First Search (DFS) tree, such that neighbouring agents appear on the same branch in the tree. Each agent a_i maintains a lower (LB_i) and upper (UB_i) bound on the cost of its subtree, which means that the lower and upper bounds of the root agent are bounds for the problem as a whole. Let H_i be the set of higher priority neighbours of a_i, and let L_i be the set of its children. During search, each agent repeatedly performs a number of tasks:

1. `VALUE` messages, containing variable assignments, are received from higher priority agents and added to the current context CC_i, which is a record of higher priority neighbours' current assignments: $CC_i \in \prod_{j:a_j \in H_i} D_j$.

[1] In this study we restrict our attention to binary inter-agent constraints, i.e. constraints do not act on more than two agents.

2. `COST` messages, containing lower and upper bounds, are received from children and stored if they are valid for the current context – for each subtree, rooted by an agent $a_j \in L_i$, a_i maintains a lower bound, $lb(l_i, a_j)$, and an upper bound $ub(l_i, a_j)$ for each of its assignments l_i. Each cost is valid for a specific context $CX(l_i, a_j) \in \prod_{k:a_k \in H_j} D_k$. Any previously stored cost with a context incompatible with the current context is reset to have lower/upper bounds of $0/\infty$.
3. A `THRESHOLD` message is received from the immediate parent of a_i – the threshold t_i is the best known lower bound for the subtree rooted by a_i.[2]
4. The local assignments with minimal lower and upper bound costs are calculated. Let C_{ij} be the constraint between x_i and x_j. The partial cost, $\delta(l)$, for an assignment of l_i to x_i is the sum of the agent's local cost $f_i(l_i)$, plus the costs of constraints between a_i and higher priority neighbours: $\delta(l_i) = f_i(l_i) + \sum_{j:a_j \in H_i} C_{ij}(l_i, CC_{i \downarrow x_j})$. The lower bound, $LB(l_i)$, for an assignment of l_i to x_i is the sum of $\delta(l_i)$ and the currently known lower bounds for all subtrees: $LB(l_i) = \delta(l_i) + \sum_{j:a_j \in L_i} lb(l_i, a_j)$. The upper bound, $UB(l_i)$, is the sum of $\delta(l_i)$ and the currently known upper bounds for all subtrees: $UB(l_i) = \delta(l_i) + \sum_{j:a_j \in L_i} ub(l_i, a_j)$. The minimum lower bound over all assignment possibilities, LB_i, is the lower bound for the agent a_i: $LB_i = min_{l_i \in D_i} LB(l_i)$. Similarly, UB_i is the upper bound for the agent a_i: $UB_i = min_{l_i \in D_i} UB(l_i)$.
5. The agent's current assignment, d_i, is updated and sent to all neighbours in L_i: if $t_i == UB_i$ then $d_i \leftarrow l_i$ that minimises $UB(l_i)$, else if $LB(d_i) > t_i$ then $d_i \leftarrow l_i$ that minimises $LB(l_i)$.
6. LB_i and UB_i are passed as costs to the parent of a_i, along with the context to which they apply, CC_i.

As the search progresses, the bounds are tightened in each agent until the lower and upper bounds are equal. If an agent detects this condition, and its parent has terminated, then an optimal solution is found and it may terminate.

To avoid exponential memory requirements, each agent stores only one set of costs for each of its possible assignments, for each of its subtrees. Whenever an agent's current context changes it checks to see if the stored costs are compatible with the new context. Incompatible costs are reset to have lower/upper bounds of $0/\infty$. If a previously visited context is returned to, then the costs for it need to be re-discovered, so there is a significant overhead incurred in context switching. By reducing context switching we can prune the search space. One method of doing this is to use informed lower bounds. Ali et al. [9] proposed a preprocessing that produces lower bounds that are then used during a subsequent execution of ADOPT. They demonstrated that if incompatible costs are reset to have non-zero lower bounds, then context switching can be reduced. While useful, the proposed

[2] The threshold in ADOPT is used to reduce thrashing. During search agents discover lower bounds for different contexts. When an agent returns to a previously explored context, the search is guided by the fact that the agent knows it cannot find an assignment with a cost better than the threshold. For a detailed explanation of thresholds and ADOPT, please refer to [4].

Algorithm 1: AdoptRelax

```
1 for relaxLevel = n − 1 to 0 do
2 |   currentProblem ← phase[relaxLevel];
3 |   ADOPT();
4 |   if relaxLevel > 0 then save();
5 |   else terminate();
```

technique is not always appropriate or efficient because: (i) each agent produces bounds for *all* of its parent's possible assignments, while in fact the parent may have private constraints or constraints with other agents eliminating some of these assignments; (ii) when an agent has multiple variables this approach requires repeatedly solving its local problem for each possible parent assignment, which can be expensive for large local problems. In Section 4, we make use of the same concept of 'informed lower bounds', but do so within a new relaxation framework.

3 Relaxation Framework for ADOPT

AdoptRelax (Algorithm 1) builds on the Adopt algorithm to allow iterated searches on a number of problem relaxations that lead to the optimal solution of the original problem. In a similar style to [8], the search is split into n phases, i.e, $n-1$ relaxations, and a final search on the original problem. The current phase is denoted by $relaxLevel$, whereby $n-1$ is the most relaxed problem and 0 is the original problem. The first phase uses the most relaxed problem. Once a solution to the current problem has been found, the relaxation level is checked (line 4). If the solution is for the original problem the algorithm terminates as normal (5). If it is for a relaxed problem, each agent will save its current lower bounds for each subtree and each assignment, including the contexts to which these bounds apply (4).[3] Once all agents have saved, the next search phase begins.

The next phase of the search has two advantages over the initial search. First, the root has a lower bound that will be propagated down through the priority tree as thresholds to each agent, preventing some repeated search. Second, each agent has a lower bound for each subtree and each of its local assignments. When costs get reset, this lower bound can be used whenever the current context is compatible with the context of the stored lower bound, resulting in reduced context switching. Using a general DisCOP algorithm such as Adopt in each search phase provides us with a general framework that allows us to compute lower bounds in a decentralised manner for arbitrary problem relaxations with different topologies. While it would also be possible to use other algorithms to

[3] By saving a single context-dependent set of bounds for each subtree and each assignment, we keep to the principles of the original Adopt algorithm, which requires polynomial space. More information could be stored, potentially leading to greater improvements, but would also lead to greater memory requirements.

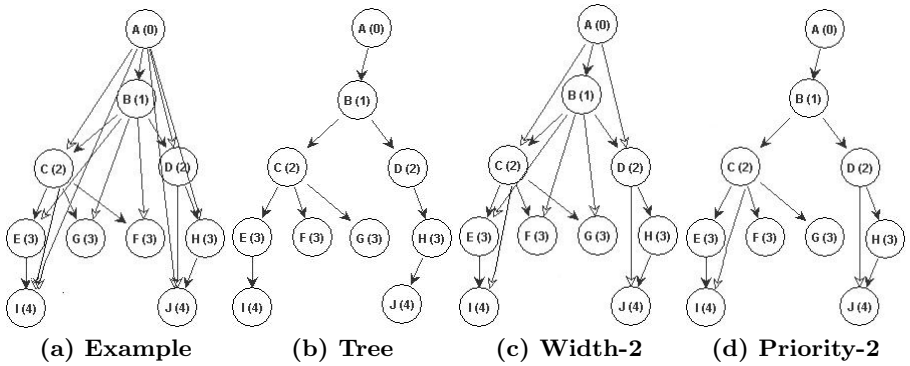


Fig. 1. (a) Example DisCOP agent graph. Arrows indicate constraints between agents, with black arrowheads indicating parent-child relationships within the priority tree. Number indicates level of agent in priority tree. (b) *TREE* relaxation removes all non-parent/child constraints. (c) *WIDTH-2* removes all constraints that span greater than 2 levels. (d) *PRIORITY-2* removes all non-parent/child constraints from top 2 levels.

solve the relaxed problems, it may be more difficult to exchange information between phases such that the information could still be used by ADOPT.

4 Relaxations in ADOPT

To use the relaxation framework we must first define problem relaxations. There are a number of different ways in which to relax distributed constraint problems, e.g. agents could be removed, constraints could be deleted or forbidden tuples could be removed from the constraints. Previous experimental analysis has shown the number of inter-agent constraints to be a key factor in determining the 'hardness' of distributed constraint problems [10, 4, 5], so we will focus on removing inter-agent constraints. The next question is deciding which constraints to remove. We want to remove constraints to produce a relaxed problem that can be quickly solved, while still containing enough of the original problem to provide meaningful lower bounds. We will use our knowledge of the context switching behaviour and the priority tree structure to determine which constraints to remove. Fig. 1.a shows the priority tree of an example problem. The current context of each agent consists of assignments to all higher priority neighbours of the agent, plus higher priority non-neighbours that impact on the costs received by the agent. We can reduce the space of possible contexts in agents, and in turn the number of context switches that will occur, by removing constraints. We now make two important observations:

1. The costs stored by an agent may become incompatible if they are dependent on agents of higher priority. E.g. the costs that agent H stores for its child J have a context that contains the assignment to D (because J has a constraint with D), and so become incompatible if D changes its assignment.

2. The higher up in the search tree that a context switch occurs, the greater the potential impact, i.e, when agents change their assignment, it will lead to a new search involving all lower priority agents, and so a context switch in higher priority agents can be more expensive than in lower priority agents. E.g. a context switch in agent B may affect all agents $C - J$, while a context switch in D only affects agents H and J.

Based on these observations, we propose three relaxations to investigate:

1. ***TREE***: remove all non-parent/child constraints in the tree;
2. ***WIDTH-X***: remove all constraints spanning more than X levels in the tree;
3. ***PRIORITY-X***: remove all non-parent/child constraints from agents with priority less than X.

Taking into account the first observation, in the *TREE* relaxation, we remove all non-parent/child links, reducing the context space of each agent to be dependent on just one other agent – its immediate parent (Fig. 1.b). In this relaxation, all costs received by an agent are independent of any higher priority agents, and so they are valid for all contexts and will never need to be reset. The *TREE* relaxation should make the problem much easier to solve, but if the network is densely connected it will remove many constraints, which means that the resulting bounds may not be good approximations. It may still be useful for loosely connected networks and also where agents have complex local problems. By only removing inter-agent constraints, each agent's internal problem is still considered in full, and so local costs still contribute to the relaxed bounds.

If we want to retain more constraints, we can generalise the *TREE* relaxation. *WIDTH-X* reduces the context space of each agent to be dependent on at most X agents (Fig. 1.c). This is done by removing all constraints that span greater than X levels in the tree, thus reducing the *width* [11] of the given graph ordering to be at most X. In fact, *TREE* = *WIDTH-1*. This relaxation allows us to trade off between solving the relaxed problem quickly (low values for X) or getting a good lower bound (high values for X). It may be that different values of X may be of use for different density networks. It should be noted that in *TREE* the lower bounds found in the relaxation are compatible with all contexts, while in *WIDTH-2* this is not the case. E.g, in Fig. 1.c, the lower bounds of agent H for its child J are dependent on the assignment of D. This means that the final bounds stored by H will be useful when solving the original problem only when D has an assignment compatible with the stored context.

Our next relaxation considers the second observation we made previously. That is, we would like to reduce context switches in agents higher up in the search tree. The *PRIORITY-X* relaxation is thus biased towards removing constraints that appear higher up in the tree. *PRIORITY-X* removes all non-parent/child constraints from agents with a priority less than X (Fig. 1.d). This may allow fewer constraints to be removed while achieving greater search savings.

Each of these relaxations provide lower bounds that can be used to prune the search space in subsequent search phases. Multiple relaxations can be used in a single execution of the algorithm, with the bounds from each phase feeding into

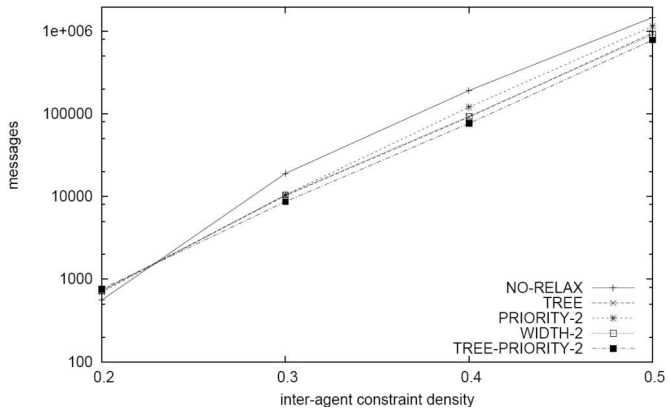


Fig. 2. Random DisCOPs varying inter-agent constraint density: 10 agents, each with one variable of domain size 5; tightness = 0.9; costs from 1–3.

the subsequent phase, e.g. *TREE* could be followed by *PRIORITY-2* before the original problem is solved. Finally, note that these relaxations can be performed in a distributed manner. The priority tree can be created using a decentralized algorithm [12]. Then, using only knowledge of their own priority and the priority of their neighbours, agents can remove the necessary inter-agent constraints.

5 Experiments

We compare the original ADOPT with ADOPTRELAX on two problem domains: random distributed constraint optimisation problems, and meeting scheduling. ADOPTRELAX is run using each of *TREE*, *WIDTH-2* and *PRIORITY-2* relaxations individually as part of a two-phase search, and also using the combination *TREE-PRIORITY-2* as part of a three-phase incremental search. The experiments are run in a simulated distributed environment: we use one machine but each agent runs asynchronously on its own thread. In problems where agents have multiple variables, a centralised solver is used to make local assignments. To compare performance, we recorded the number of messages communicated by the agents, and also the number of Non-Concurrent Constraint Checks (NCCC) [13]. The results of both metrics were comparable, so we now only display graphs for the number of messages. All results are averaged over 20 test instances.

In the random problems, we use 10 agents, each with a single variable of domain size 5. For each constraint, 90% (the tightness) of tuples have a non-zero cost chosen uniformly from the set $\{1, 2, 3\}$. The inter-agent constraint density is varied between 0.2 and 0.5.[4] A characteristic of these problems is that all costs are on inter-agent constraints, i.e. there are no local agent costs. Fig. 2 (log scale) shows that the relaxations give an improvement over the standard ADOPT as the

[4] Increasing the number of inter-agent constraints is expensive [10]. Most DisCOP algorithms are tested on problems with densities no greater than 0.4, e.g. [4, 2, 14].

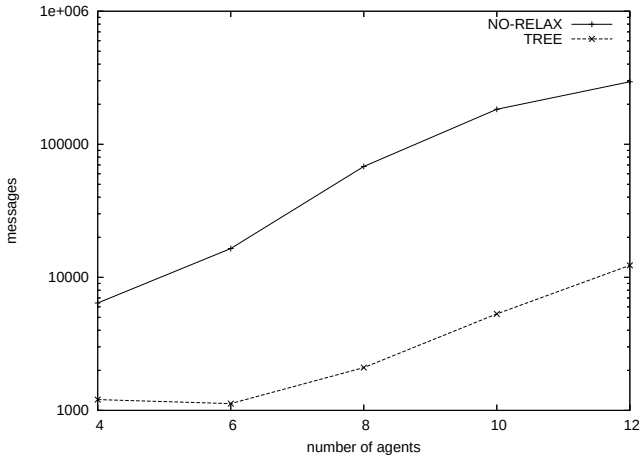


Fig. 3. Meeting scheduling problems: number of meetings = number of agents; 2 attendees per meeting; 2 personal tasks per agent; maximum of 4 meetings per agent.

density increases. *PRIORITY-2* finds high lower bounds and for less dense problems these bounds are found quickly. For denser problems it can take longer, and so the benefit from relaxation only accrues late in the search, hence worse performance for higher densities. *TREE* always finds bounds quickly, although for denser problems, these bounds will be further from the actual solution. *TREE* slightly outperforms *WIDTH-2* up to a density of 0.4 but *WIDTH-2* is better for 0.5. By combining two relaxations, *TREE-PRIORITY-2*, there is an increased overhead. However, as the density increases, this overhead becomes less important and *TREE-PRIORITY-2* outperforms the other relaxations, giving almost 50% improvement over ADOPT for density 0.5.

We generate meeting scheduling problems following a model used by [2] and others. For each meeting each agent is involved in, it owns a variable with 8 values (meeting starting times). Variables in different agents that represent the same meeting are linked with equality constraints. Agents also have personal tasks (single variables with 8 values). Variables in the same agent are linked with inequality constraints (the agent can not have two meetings/tasks at the same time). Agents have preferences, represented as costs, for the values they would like to assign to each meeting/task. Note in these problems the inter-agent constraints are hard constraints, which will have a cost of 0 when satisfied. Therefore, the costs incurred in solutions to the problem are local costs, i.e. the preferences of the agents. Removing inter-agent constraints allows the agents to choose more preferable local assignments, but in both the relaxed and original problems local costs will be incurred.

In our first experiment, we set the number of meetings equal to the number of agents. This setting means that all agent graphs will be a tree plus one additional constraint. Relaxing this constraint using the *TREE* relaxation gives remarkable results (Fig. 3), saving over an order of magnitude reduction in messages. The

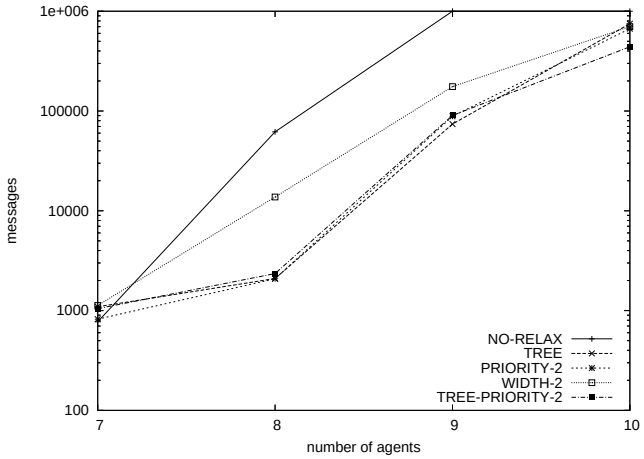


Fig. 4. Results of meeting scheduling problems: inter-agent link/meeting density = 0.3; 2 attendees per meeting; 2 personal tasks per agent; max. 4 meetings per agent.

key reason for this is that the agents have significant local costs and since the problem relaxations consider the local problems in full, strong lower bound approximations can be found, allowing greater pruning of the search space. In Fig. 4 we show results for increasing the number of agents with the inter-agent constraint density fixed to 0.3. All relaxations apart from *WIDTH-2* show over an order of magnitude improvement for 8 agents, and ADOPT hits an imposed cutoff of 10^6 messages for all instances greater than 8 agents. *PRIORITY-2* and *TREE* are successful for lower numbers of agents, but *TREE-PRIORITY-2* is the best once the problems are increased to contain 10 agents. *WIDTH-2* becomes more competitive when there are more agents and more opportunities to remove constraints.

Note that although we achieve very good results in these problem domains, care should be taken in applying these relaxation techniques. If most of the costs in the problem are incurred by, or are dependent on, inter-agent constraints, then removing these constraints may produce extremely poor lower bounds, and thus the general graph-based relaxation methods presented here may be counterproductive. In such cases, relaxation heuristics that take account of the structure of the objective function would be required.

6 Conclusions and Future Work

We have proposed ADOPTRELAX, a novel relaxation framework that is an extension of the ADOPT DisCOP algorithm. ADOPTRELAX allows an arbitrary number of problem relaxations to be solved prior to solving the original problem. These relaxations produce lower bounds that allow portions of the search space to be pruned. We have proposed a number of graph-based relaxations which remove inter-agent constraints. We have shown, through experimental analysis

on random DisCOPs and meeting scheduling that AdoptRelax can offer an order of magnitude speed-up, particularly where agents have significant local costs. Future work will investigate alternative relaxation heuristics, as well as examining relaxations where the constraints are modified rather than removed. We will also investigate if algorithms other than Adopt could be useful when solving the relaxations.

References

1. Béjar, R., Domshlak, C., Fernàndez, C., Gomes, C., Krishnamachari, B., Selman, B., Valls, M.: Sensor Networks and Distributed CSP: Communication, Computation and Complexity. Artificial Intelligence **161**(1-2) (2005) 117–147
2. Petcu, A., Faltings, B.: A Scalable Method for Multiagent Constraint Optimization. In: Proc. 19th Int. Joint Conference on Artificial Intelligence. (2005) 266–271
3. Yokoo, M., Hirayama, K.: Algorithms for Distributed Constraint Satisfaction: A Review. Autonomous Agents and Multi-Agent Systems **3**(2) (2000) 185–207
4. Modi, P., Shen, W., Tambe, M., Yokoo, M.: ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. Artificial Intelligence **161**(1–2) (2005) 149–180
5. Burke, D., Brown, K.: Efficient Handling of Complex Local Problems in Distributed Constraint Optimization. In: Proc. 17th European Conference on Artifical Intelligence. (2006) 701–702
6. Hirayama, K., Yokoo, M.: An Approach to Over-constrained Distributed Constraint Satisfaction Problems: Distributed Hierarchical Constraint Satisfaction. In: Proc. 4th International Conference on Multi-Agent Systems. (2000) 135–142
7. Marinescu, R., Dechter, R.: AND/OR Branch-and-Bound for Graphical Models. In: Proc. 19th Int. Joint Conference on Artificial Intelligence. (2005) 224–229
8. Sacerdoti, E.D.: Planning in a Hierarchy of Abstraction Spaces. Artificial Intelligence **5**(2) (1974) 115–135
9. Ali, S.M., Koenig, S., Tambe, M.: Preprocessing techniques for Accelerating the DCOP Algorithm ADOPT. In: Proc. 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems. (2005) 1041–1048
10. Hirayama, K., Yokoo, M., Sycara, K.: The Phase Transition in Distributed Constraint Satisfaction Problems: First Results. In: Proc. 6th International Conference on Principles and Practice of Constraint Programming. (2000) 515–519
11. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
12. Chechetka, A., Sycara, K.P.: A Decentralized Variable Ordering Method for Distributed Constraint Optimization. In: Proc. 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems. (2005) 1307–1308
13. Meisels, A., Razgon, I., Kaplansky, E., Zivan, R.: Comparing Performance of Distributed Constraints Processing Algorithms. In: Proc. 3rd International Workshop on Distributed Constraint Reasoning. (2002) 86–93
14. Gershman, A., Meisels, A., Zivan, R.: Asynchronous forward-bounding for distributed constraints optimization. In: Proc. 17th European Conference on Artifical Intelligence. (2006) 103–107

Automated Constraint Reformulation for Explanation

Hadrien Cambazard and Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{h.cambazard|b.osullivan}@4c.ucc.ie

Abstract. Many approaches to explanation generation in constraint satisfaction problems have been reported in the literature. The dominant approach is based on computing minimal conflicting sets of constraints. An explanation can be considered concise if it involves constraints of low arity over a small subset of the variables of the problem. However, in many practical domains constraints are specified extensionally as tables of allowed assignments of values to variables. Such tables often have high arity. From an explanation point of view, table constraints can prevent us from finding concise explanations. In this paper we present an approach to automatically reformulate large arity table constraint into a set of low arity constraints whose conjunction is logically equivalent to the original table. We demonstrate the utility of our approach on a number of real-world table constraints from the fields of product configuration and machine learning.

1 Introduction

Constraint satisfaction techniques are ubiquitous in many practical problem-solving contexts. The fundamental notion in constraint satisfaction is to separate the representation of a problem from the method used to solve it. As the number of interactive systems built upon constraints technology increases, several issues become a concern. For example, how should we integrate information sources that are more naturally represented as tables, or simple databases, with more traditional constraint-based methods? Also, how can we support the generation of concise explanations when a set of constraints cannot be satisfied? It is common in user-focused interactive applications to allow users to add unary constraints, i.e. assignments to variables, to a set of background constraints that define the general problem being solved. When such a set of constraints cannot be satisfied, one can compute a set-wise maximal subset of the user's choices that are consistent, or a set-wise minimal subset of his constraints that are inconsistent. These are the standard notions of maximal (minimal) relaxation (conflict) that have been used in the artificial intelligence community [4, 7]. In the context of general arity constraints, computing a minimal conflict can be more challenging. However, again, there is a significant literature on finding minimal conflicting sets of constraints, sometimes referred to as minimal unsatisfiable subproblems [8]. The duality between conflicting sets of constraints and relaxations is also well known [2, 3, 7, 10].

In this paper we address a novel question in this area: how do we compute *concise* explanations when we have problems involving constraints of very high arity, i.e. defined over many variables in the problem. We regard an explanation as concise if it

involves constraints of low arity over a small subset of the variables of the problem. In particular, we consider the case in which some constraints are defined using tables of consistent assignments to a set of variables. We use techniques used to normalise relational databases to automatically reformulate a large arity constraint into a conjunction of lower arity constraints. We can guarantee that the set of solutions to the reformulation is equivalent to that of the original set of constraints. Therefore, using the reformulation, we can find explanations that involve constraints of much lower arity over a smaller set of variables.

The remainder of the paper is organised as follows. In Section 2 we present the formal background required throughout the paper. We demonstrate the utility of constraint reformulation for generating explanations in Section 3. We present our approach to constraint reformulation in Section 4. An empirical evaluation is presented in Section 5.

2 Background

A constraint satisfaction problem (CSP) is defined by a set of variables, each of which must be assigned a value from its domain, subject to a set of constraints. Each constraint restricts the set of consistent assignments to a subset of the variables.

Definition 1 (Constraint Satisfaction Problem). *A constraint satisfaction problem $\mathcal{P}$ is a triple $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where: $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of variables; $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$ is a set of domains corresponding to the possible values of each variable; and $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of m constraints. Each constraint $c_i \in \mathcal{C}$ is defined by a pair $\langle scope(c_i), rel(c_i) \rangle$, where $scope(c_i)$ denotes an ordered subset of $\mathcal{X}$, and $rel(c_i)$ is a set of tuples over $scope(c_i)$ that satisfy the constraint c_i. The number of variables constrained by c_i, i.e. $|scope(c_i)|$, is known as the* arity *of the constraint c_i.*

Definition 2 (Solution to a CSP). *Given a CSP $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ and given any constraint $c_i \in \mathcal{C}$, a labelling t to the variables in $scope(c_i)$ satisfies c_i if $t \in rel(c_i)$. We denote as $t[x_i]$ the value $v \in D(x_i)$ assigned to x_i in t. A labelling t of $\mathcal{X}$ is a solution of $\mathcal{P}$ if for every $c_i \in \mathcal{C}$, the restriction of t to $scope(c_i)$ satisfies c_i. We denote as $sol(\mathcal{P})$ the set of all solutions to $\mathcal{P}$.*

Example 1 (A Simple CSP). We exemplify the definitions given above on a simple example CSP in which $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, $\mathcal{D} = \{D(x_1) = \{0, 1, 2\}, D(x_2) = \{0, 2, 4\}, D(x_3) = \{0, 1, 2, 3\}, D(x_4) = \{2, 3, 4\}\}$, and the constraints $\mathcal{C}$ are:

$$
\begin{aligned}
c_1(x_1, x_3) &\stackrel{\text{def}}{=} \{(0,0), (1,0), (2,1), (0,2), (2,3)\}, \\
c_2(x_1, x_4) &\stackrel{\text{def}}{=} \{(0,4), (1,2), (2,3), (2,2)\}, \\
c_3(x_2, x_3) &\stackrel{\text{def}}{=} \{(0,0), (4,1), (4,2), (2,3)\}, \\
c_4(x_2, x_4) &\stackrel{\text{def}}{=} \{(0,4), (0,2), (4,3), (4,4), (2,2)\}.
\end{aligned}
$$

The constraints scopes are $\{x_1, x_3\}$, $\{x_1, x_4\}$, $\{x_2, x_3\}$, and $\{x_2, x_4\}$, and the relations are the sets specified in the right-hand sides above. The set of solutions are as follows

$$\{(0,0,0,4), (0,4,2,4), (1,0,0,2), (2,2,3,2), (2,4,1,3)\}$$

using the natural subscript ordering for the variables in $\mathcal{X}$. ▲

In this paper we are interested in automatically computing *equivalent* reformulations of large arity table constraints. It is useful, therefore, to formally define the meaning of equivalence between CSPs.

Definition 3 (Equivalent CSPs). *Given a CSP* $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ *and a CSP* $\mathcal{P}' \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \mathcal{C}' \rangle$ *we say that* $\mathcal{P}$ *is equivalent to* $\mathcal{P}'$ *if both problems have equivalent sets of solutions, i.e.* $sol(\mathcal{P}) \equiv sol(\mathcal{P}')$.

When we reformulate a high arity constraint c_i we obtain a conjunction of lower arity constraints, $c_i^1, \ldots, c_i^k$, where the scope of each c_i^j is a subset of $scope(c_i)$. To define the notion of restricting a tuple of allowed values to a sub-scope of the original constraint we use the projection operator from Codd's Relational Algebra.

Definition 4 (Scope Restriction [5]). *Let* r *be a relation instance over a set of variables* Y *and* t*, a tuple of* r*. The projection onto* $Z \subseteq Y$ *of* t*, denoted* $t[Z]$*, is the restriction of* t *to* Z*. The projection of* r *onto* Z*, denoted* $\sigma_Z(r)$*, is the set* $\{t[Z] | t \in r\}$.

Scope restriction is a form of constraint relaxation. Informally, by restricting the scope of a constraint we are reducing the number of variables that are constrained. Therefore, any tuple satisfying the original constraint also satisfies the relaxation.

Definition 5 (Relaxation of a Positive Table Constraint). *We say that constraint* c_i^r *with a scope* $X = scope(c_i^r)$ *is a relaxation of constraint* c_i *if* $X \subseteq scope(c_i)$ *and* $rel(c_i^r) = \sigma_X(rel(c_i))$*, i.e. the set of allowed tuples of the relaxation* c_i^r *are obtained by projecting the set of allowed tuples of* c_i *onto the set of variables constrained by* c_i^r*. The constraint* c_i^r *corresponding to the relaxation of* c *on* X *is also denoted* $\Pi_X(c_i)$.

Based on the notion of constraint relaxation, we can regard a reformulation of a constraint as a subset of its (irredundant) relaxations.

Definition 6 (Constraint Reformulation). *A reformulation* $\Delta(c_i)$ *of a positive table constraint* c_i *is a set of relaxations* $\mathcal{R} \stackrel{\text{def}}{=} \{c_i^1, \ldots, c_i^k\}$ *of* c_i *such that* $\forall c_i^x, c_i^y \in \mathcal{R}, x \neq y$*,* $scope(c_i^x) \not\subseteq scope(c_i^y)$ *and* $scope(c_i^y) \not\subseteq scope(c_i^x)$.

As a notational convenience, we often denote a reformulation of a constraint c_i as a set of subsets of $scope(c_i)$ defining the relaxations involved.

Example 2 (Constraint Reformulation). Consider the following constraint, over the same variables and domains as those in Example 1:

$$c_a(x_1, x_2, x_3, x_4) \stackrel{\text{def}}{=} \{(0,0,0,4), (0,4,2,4), (1,0,0,2), (2,2,3,2), (2,4,1,3)\}.$$

A reformulation of this constraint, in terms of two relaxations of c_a, is as follows:

$$\begin{aligned} c_a^1(x_1, x_2, x_4) &\stackrel{\text{def}}{=} \{(0,0,4), (1,0,2), (2,4,3), (0,4,4), (2,2,2)\}, \\ c_a^2(x_1, x_3) &\stackrel{\text{def}}{=} \{(0,0), (0,2), (1,0), (2,1), (2,3)\}. \end{aligned}$$

We will refer to this reformulation as $\Delta(c_a) = \{\{x_1, x_2, x_4\}, \{x_1, x_3\}\}$. Note that the set of solutions to the reformulation is not equivalent to those of c_a, since the tuple $(0,4,0,4)$ is allowed by this reformulation. However, note that the set of constraints in Example 1 forms an equivalent reformulation of c_a. ▲

Reformulations that give rise to conjunctions of constraints that are equivalent, i.e. have the same set of solutions, are very important. We refer to such reformulations as *lossless*, in keeping with the standard terminology in databases[1].

Definition 7 (Lossless Reformulation). *Given a CSP* $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \{c_i\} \rangle$ *involving a single constraint* c_i. $\Delta(c_i)$ *is a lossless reformulation of* c_i *if the CSP* $\mathcal{P}' \stackrel{\text{def}}{=} \langle \mathcal{X}, \mathcal{D}, \Delta(c_i) \rangle$ *is such that* $sol(\mathcal{P}) \equiv sol(\mathcal{P}')$.

3 An Application of Lossless Constraint Reformulation

In many real-world problems we have to deal with large arity table constraints. For example, in the well-known Renault Megane Car Configuration Problem [1], all constraints are represented as extensional table constraints defined by a set of allowed tuples of assignments. Most constraints in this problem have arities between five and nine variables. When a set of constraints does not admit a solution, a common form of explanation is to generate a minimal conflicting set of constraints, i.e. a set of constraints that is inconsistent, but all subsets of this set are consistent. When our problem formulation contains many high arity constraints, such an explanation may not be concise. However, a lossless reformulation may exist that gives a very concise explanation.

Example 3 (Explanations from Lossless Reformulations). Consider the following set of constraints, each of arity four:

$$c_1(x_1, x_3, x_4, x_6) \stackrel{\text{def}}{=} \{(2,0,0,1), (1,1,1,2), (2,0,2,1), (1,1,0,0)\},$$
$$c_2(x_1, x_2, x_5, x_7) \stackrel{\text{def}}{=} \{(0,1,0,1), (1,2,2,0), (1,0,1,0), (2,1,0,2)\},$$
$$c_3(x_5, x_2, x_3, x_4) \stackrel{\text{def}}{=} \{(1,0,2,2), (2,1,1,2), (0,2,0,1), (2,1,2,1)\}.$$

This set of constraints does not admit a solution. A minimal conflict set that is sufficient to explain this inconsistency involves all three constraints. This is because by removing any of the constraints, the problem becomes consistent and a solution can be found.

However, a lossless reformulation provides a much more compact explanation. It can be shown that each of the constraints above is equivalent to the corresponding conjunctions of constraints given below:

$$c_1(x_1, x_3, x_4, x_6) \equiv \{c_1^1(x_1, x_3), c_1^2(x_1, x_6), c_1^3(x_4, x_6)\},$$
$$c_2(x_1, x_2, x_5, x_7) \equiv \{c_2^1(x_1, x_7), c_2^2(x_5, x_2), c_2^3(x_1, x_2)\},$$
$$c_3(x_5, x_2, x_3, x_4) \equiv \{c_3^1(x_2, x_5), c_3^2(x_2, x_3, x_4)\}.$$

On this reformulation, the inconsistency can be more compactly explained by using the following constraints from the reformulation:

$$\{c_1^1(x_1, x_3), c_3^2(x_2, x_3, x_4), c_2^3(x_1, x_2)\}.$$

In fact, an even more compact explanation can be found since in c_3^2, x_4 is not required to explain the inconsistency. However no lossless reformulation can discover this. ▲

We now present how lossless reformulations can be computed automatically.

[1] Note that in our context by selecting a reformulation that is not lossless, the reformulation admits solutions that are inconsistent with the original constraint. In this setting we have "lost" information ruling out some combinations of values.

4 Reformulation of Positive Table Constraints

In contrast with earlier work, e.g. [5], our approach to computing lossless reformulations of positive table constraints exploits the concept of functional dependencies in a relation [6]. A *functional dependency* in a relation $rel(c_i)$ is written as $\mathcal{F}_i : X_i \rightarrow y_i$, where $X_i \cup \{y_i\} \subseteq scope(c_i)$. A functional dependency states that if a pair of tuples in the relation take the same values for the variables in X_i, they must also take the same value for variable y_i. A functional dependency $\mathcal{F}_i : X_i \rightarrow y_i$ is minimal if y_i is not functionally dependent on any subset of X_i. A dependency $\mathcal{F}_i : X_i \rightarrow y_i$ is said to be *trivial* if $y_i \in X_i$. Algorithms for finding the set of all minimal and non-trivial dependencies that hold in a given relation are known [6].

Example 4 (Functional Dependencies in Constraint Relations). Consider constraint c_a from Example 2, presented below:

$$c_a(x_1, x_2, x_3, x_4) \stackrel{\text{def}}{=} \{(0,0,0,4), (0,4,2,4), (1,0,0,2), (2,2,3,2), (2,4,1,3)\}.$$

The following minimal functional dependencies (among the seven that exist for this relation) hold in c_a: $\mathcal{F}_1 : \{x_3\} \rightarrow x_2, \mathcal{F}_2 : \{x_1, x_2\} \rightarrow x_3$, and $\mathcal{F}_3 : \{x_1, x_2\} \rightarrow x_4$. The values of x_2 are uniquely determined by the value of x_3 and the values of x_4 and x_3 depend, similarly, only on the values taken by x_1 and x_2. The dependency $\{x_2\} \rightarrow x_3$ does not hold because value 4 of x_2 does not determine the value of x_3 (2 or 1). ▲

Based on a set of functional dependencies that hold on the relation of a constraint, we define a lossless reformulation of the constraint into a conjunction of constraints as follows. Given a positively defined table constraint c_i and a functional dependency $\mathcal{F}_i : X_i \rightarrow y_i$ holding on $rel(c_i)$, constraint c_i can be reformulated into a pair of constraints. The first constraint is defined over the scope $scope(c_i) - \{y_i\}$, while the second constraint is defined over the scope $X_i \cup \{y_i\}$. Informally, a functional dependency allows us to split the scope of a constraint by eliminating the functionally dependant variable y_i.

Example 5 (Constraint Reformulation using a Functional Dependency). Consider the constraint c_a and the functional dependencies presented in Example 4. The original scope of c_a is (x_1, x_2, x_3, x_4). If we apply $\mathcal{F}_3 : \{x_1, x_2\} \rightarrow x_4$, this scope is split into (x_1, x_2, x_3) and (x_1, x_2, x_4), according to the procedure above. If we apply $\mathcal{F}_1 : \{x_3\} \rightarrow x_2$ on the scope (x_1, x_2, x_3) we can split this into (x_1, x_3), (x_2, x_3) and the resulting lossless reformulation of c_a is made of (x_1, x_3), (x_2, x_3) and (x_1, x_2, x_4). ▲

Since we perform constraint reformulation using functional dependencies the following theorem immediately follows.

Theorem 1 (Lossless Reformulation). *The reformulation $\Delta(c_i)$ of constraint c_i using the functional dependency $\mathcal{F}_i : X_i \rightarrow y_i$ holding on $rel(c_i)$ is lossless.*

A reformulation is obtained by applying a sequence of dependencies. However, as dependencies are applied, others may no longer be applicable to the reformulation we obtain. For example, in Example 5 we did not apply functional dependency $\mathcal{F}_2$:

$\{x_1, x_2\} \rightarrow x_3$. This is because having applied $\mathcal{F}_1$ and $\mathcal{F}_3$ it is not longer applicable, since no scope in our reformulation can be decomposed with it. Two dependencies cannot be applied in the same sequence if they have, for example, the same right-hand side. Others have to be applied in a given order, due to the following theorem.

Theorem 2 (Valid Ordering of Functional Dependencies). *Given a constraint c_i, let $\mathcal{F}_i : X_i \rightarrow y_i$ and $\mathcal{F}_j : X_j \rightarrow y_j$ be two minimal functional dependencies that hold in $rel(c_i)$ such that $y_j \in X_i \cup \{y_i\}$ and $X_i \cup \{y_i\} \not\subset X_j \cup \{y_j\}$. Then $\mathcal{F}_i$ can only be applied before $\mathcal{F}_j$, which we denote as $\mathcal{F}_i \prec \mathcal{F}_j$.*

Therefore, a set of functional dependencies can be viewed as a directed graph in which each vertex represents a functional dependency. A directed edge $(\mathcal{F}_i, \mathcal{F}_j)$ in this graph between the vertices $\mathcal{F}_i$ and $\mathcal{F}_j$ is added if $\mathcal{F}_i$ *can only be applied before* $\mathcal{F}_j$. Provided the set of functional dependencies are applied in an order that does not correspond to a cyclic path in this graph, the resultant reformulation is unique.

Given a set of functional dependencies, we seek to use them to find a reformulation in which the maximum arity of the constraints in the reformulation is minimised. This problem is NP-Hard, since the corresponding decision problem can be used to solve the Feedback Vertex Set problem, which is known to be NP-Complete. However, our experiments show that, even on real-world table constraints, the optimal reformulation can be found in less than one second.

5 Experiments

The objective of our experimental evaluation was twofold[2]. Firstly, we wished to evaluate the extent to which functional dependencies could be used to effectively reformulate real-world table constraints in order to significantly reduce the maximum arity of the constraints in the reformulation. Secondly, we sought to show that using a reformulation of table constraints based on functional dependencies, more compact minimal conflict explanations could be found. Our experiments achieved each of these objectives and support our claim that exploiting functional dependencies in constraint relations is a promising approach to automated reformulation of table constraints.

We have used a well-known library, called TANE [6], to compute the set of functional dependencies in a constraint relation. Four datasets were used in our experiments:

1. From the Renault Megane Car Configuration Problem [1] we used the two largest table constraints (R80 and R140). R80 involves 10 variables and contains 342 tuples, while R140 involves 9 variables and contains 164 tuples.
2. We used a dataset of digital cameras, defined using 11 variables and 113 tuples [9].
3. We used a dataset of laptops, defined using 13 variables and 403 tuples [9].
4. We used the CBR travel case base, defined using 9 variables and 1470 tuples.

For each dataset we removed any field that gave a unique identifer to each tuple, since these would have introduced many "artificial" dependencies. Specifically, this resulted in the arities of the digital camera and laptop data-sets being reduced to 8 and 10 variables, respectively. All other data-sets remained unchanged.

[2] Experiments were implemented using Choco (`http://choco-solver.net`), and run on a MacBook with 2Gb of RAM and a dual core 2Ghz processor.

5.1 Experiment 1: Reformulation of Real-World Constraints

The objective of this experiment was to evaluate the extent to which a functional dependency-based reformulation could be used to reformulate the table constraints in each of our data-sets. This experiment involved computing the set of all functional dependencies, and computing a reformulation with the smallest maximum arity. We summarise our results in Table 1. We report the number of tuples and arity of the initial tables. We also present the number of functional dependencies found by TANE, from which we find the best reformulation, i.e. the one that gives a reformulation whose maximum arity is smallest. For each dataset we report the number of constraints, minimum and maximum arity, and the time (in seconds) taken to compute the optimal reformulation.

Table 1. Lossless reformulations of the table constraints in our dataset.

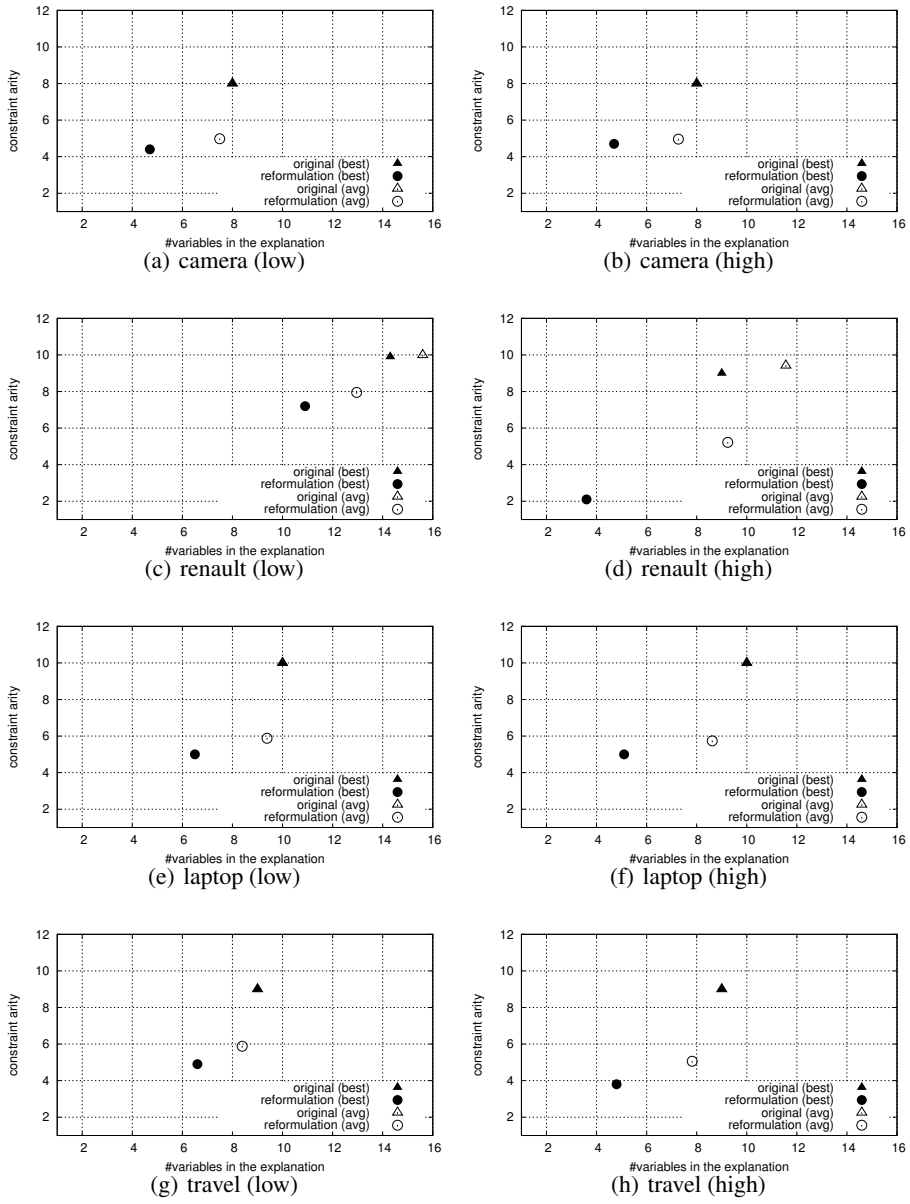
Data-set	#tuples	arity	#dependencies	#constraints	min.arity	max.arity	time(s)
camera	113	8	41	4	5	5	0.20
laptop	403	10	54	4	5	6	0.57
renault R80	342	10	2	3	2	8	0.00
renault R104	164	9	11	6	2	4	0.02
travel R0	1470	9	7	4	4	6	0.00

There are several very positive results in this table. Firstly, the optimal reformulation can be found extremely quickly, eventhough this task is theoretically intractable. Secondly, it is possible to find reformulations in which the maximum arity is significantly reduced, e.g. by 55% in the case of the Renault R104 constraint. Thirdly, it is interesting to see that the reformulations found can sometimes contain constraints of very low arity. Specifically, for both constraints from the Renault configuration problem we were able to find an equivalent reformulation involving binary constraints.

5.2 Experiment 2: Computing Compact Explanations

The objective of this experiment was to measure the compactness of the explanations of inconsistency found for each dataset using the original constraint, as well as a reformulation with the smallest maximum arity. We used the same datasets as before, but in the case of the Renault tables we combined both together in the same problem, giving us four scenarios in total. The key measurements taken in this experiment were the number of variables, along with the maximum arity of the constraints involved in the explanations of inconsistency. A more compact explanation involves fewer variables and constraints of lower arity.

The approach we adopted was, for each dataset, to add a set of randomly generated binary constraints to force inconsistency with a given (set of) table constraint(s), but are consistent without the table constraint. Therefore, in the case where the table constraints in our datasets were used in their original form, the table constraints were involved in all explanations of inconsistency. However, using the reformulation, it was often possible to find a much more compact explanation, as our results illustrate.



(a) camera (low) (b) camera (high)

(c) renault (low) (d) renault (high)

(e) laptop (low) (f) laptop (high)

(g) travel (low) (h) travel (high)

Fig. 1. Experimental results showing the compactness of the explanations found using a reformulation of a table constraint, as compared with the original case. Note that the `avg` and `best` values for the original constraint always coincide.

To generate a variety of classes of inconsistent problems for which we computed explanations, we varied both the density of constraints to be combined with the constraints in our dataset, and their tightness. The *density of a set of constraints* is the percentage of all possible binary constraints that can be defined on a set of variables that are present in the constraint network. The *tightness* of a constraint is the percentage of all possible tuples that can be assigned to a pair of variables that violate the constraint.

Table 2. Constraint density and tightness settings used in our experiments.

Dataset	density (%)	low tightness (%)	high tightness (%)
camera	43	75	88
laptop	27	69	90
travel	33	72	94
renault	8	42	88

For each dataset, 20 problems are generated: 10 with a low tightness and 10 with high tightness. The density and tightness of the random binary problems, therefore, varied for each table as it is related to the size of the initial scope and the initial domains of the variables in the constraint. The *low tightness* setting was chosen to be the first tightness found before the problems generated began to be inconsistent and the *high tightness* setting is the last tightness found before all problems generated are inconsistent. Table 2 summarises the parameters used to generate the set of additional binary constraints used for each dataset to generate inconsistencies.

To fairly compare the compactness of explanations found using the reformulation against that of the original constraint, for each inconsistent problem we computed the set of *all minimal conflicts* using a well-known algorithm by Bailey and Stuckey [2]. Based on the set of all possible minimal conflicts, we can measure either the smallest maximum arity and the total number of variables in that best explanation (we refer to this as `best`), or the average maximum arity and average total number of variables (which we refer to as `avg`).

In Figure 1 we present our results. For each dataset we show two figures: one for the cases where inconsistency was caused by the addition of low tightness constraints, while in the other high tightness constraints were used. In each case, we plot: (a) a point at a coordinate given by the average minimum number of variables in the explanation and the corresponding average smallest maximum arity (the `best` case); and (b) a point at a coordinate given by the average number of variables in the explanation and the corresponding average arity (the `avg` case). The former represents the best case, while the latter represents the average measurements for the explanations we compute. We also plot coordinates corresponding to the original constraint in each case.

It is clear that, in each case, the minimal set of constraints sufficient to explain inconsistency in the case of the reformulation is always more compact than if the original table was used. It is often possible to either find explanations that either reduce the arity of the largest constraint, or the number of variables involved in the explanation by almost half. When the constraints added are of high tightness, in the case of the Renault

tables (Figure 1(d)), the number of variables can be reduced from nine to almost three, while the maximum arity constraint in the explanation is reduced from nine to two.

This is a very significant improvement in explanation compactness. This trend is similar for all datasets. Finally, even when focusing on the performance of the average-case results, rather than the best-case, the improvements are also significant.

6 Conclusion

We have presented a novel approach to automatically reformulating constraints defined as a table of allowed assignments to variables. Constraints of this form are common in a variety of settings. We demonstrated that by using functional dependencies from the field of database design, reformulations of table constraints could be found that yield compact explanations of inconsistency by reducing both the number of variables required to explain inconsistency and the arity of the largest constraint involved in the explanation. We demonstrated our approach on real-world datasets with very positive results.

Acknowledgements

This work was supported by Science Foundation Ireland (Grant Number 05/IN/I886). We thank Pearl Pu for providing the laptop and camera datasets.

References

1. Jérôme Amilhastre, Hélène Fargier, and Pierre Marguis. Consistency restoration and explanations in dynamic CSPs – application to configuration. *Artif. Intell.*, 135:199–234, 2002.
2. James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Proceedings of PADL*, pages 174–186, 2005.
3. Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, 1987.
4. Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.
5. Marc Gyssens, Peter Jeavons, and David A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artif. Intell.*, 66(1):57–89, 1994.
6. Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.
7. Ulrich Junker. QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.
8. Mark H. Liffiton and Karem A. Sakallah. On finding all minimally unsatisfiable subformulas. In *SAT*, pages 173–186, 2005.
9. James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In *ACM Conference on Electronic Commerce*, pages 114–123, 2007.
10. Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

Experiments in Mobile Content Enrichment

Karen Church and Barry Smyth

Adaptive Information Cluster,
Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland
{Karen.Church, Barry.Smyth}@ucd.ie

Abstract. Mobile content, by its concise nature, offers limited indexing opportunities, which makes it difficult to build high-quality mobile search engines and indexes. In this paper we address this problem by evaluating a heuristic content enrichment framework that uses standard Web resources as a source of additional indexing knowledge. We present an evaluation using a mobile news service that demonstrates significant improvements in search performance compared to a benchmark mobile search engine.

1 Introduction

Recent positive developments in handsets, infrastructure, content quality, and charging models are fueling renewed optimism about the potential of the Mobile Internet. For example, a recent report by Informa put the number of mobile subscribers at 2.7 billion at the end of 2006 [1]. Ipsos Insight published a study in 2006 which shows that 28% of mobile subscribers worldwide have browsed the Internet using their mobile phone [2]. This pattern of growth was driven primarily by older users (age 35+) indicating that the traditional early adopter segment, i.e. young males, no longer dominates wireless Internet access.

Until recently, the Mobile Internet, for the majority of subscribers, has meant the "walled-garden" of content that has been available through their operator's portal. However, as off-portal content has grown so too has the interest of users in accessing this content, which has led to a sharp increase of interest in the potential for mobile search engines to provide users with fast and efficient access to this content. For example, major players within the search engine industry like Google and Yahoo continue to release exciting new improvements to their mobile search services. One recent study reports that 31% of users started using mobile search in 2006 with another 48% expecting to start in the next few months [3].

[1] IT Week: *Mobile industry 'bullish' for the new year*, http://www.itweek.co.uk/vnunet/analysis/2171436/mobile-industry-bullish-2007

[2] Ipsos Insight: *Mobile phones could soon rival the PC as worlds dominant Internet platform*, http://www.ipsos-na.com/news/pressrelease.cfm?id=3049

[3] Mobile Marketing Association: *Mobile Marketing Association announces Mobile Search study key findings*, http://www.primezone.com/newsroom/news.html?d=103374

There are many significant issues to be addressed if current search engine technologies are to deliver the type of search experience that is necessary to engage mobile users. For a start there are the many well-documented challenges of delivering information to small screen devices [5]. In addition, restricted text input capabilities inevitably have a significant impact on the use of these handsets as search devices, limiting the type of queries that will likely be provided. These device limitations are but one aspect of the larger problem and the nature of mobile content itself is such that it introduces additional challenges from an indexing and search standpoint. Mobile pages are typically much shorter in length than their Web counterparts; most mobile gateways and handsets are limited by the size of the pages they can process, and traditionally WML (Wireless Markup Language, the mobile equivalent of HTML) decks are restricted to just a few KB. Mobile Internet access is mainly about *content snacking* but smaller pages mean that there is less content for a search engine to index and thus less information available to inform retrieval. This exacerbates the so-called *vocabulary gap* [4] that plagues Web search — referring to the tendency for searchers to often chose query terms that do not correspond to those used to index their target document — because there are even fewer terms available for indexing thus reducing the likelihood of a query match during future searches.

We consider this issue of limited page content in this paper and describe a heuristic *context enrichment* strategy to extend the indexing knowledge that is used to characterise mobile content by discovering additional indexing terms. To do this for some item of content we automatically transform the content item in to a set of *enrichment queries* which are used to retrieve a set of *enrichment results* from a standard Web search engine. These related result pages then act as a source of *enrichment terms* which are extracted, ranked and added to the mobile index. We have previously described and evaluated a basic content enrichment (CE) framework, focusing on the selection of enrichment terms, and demonstrating a significant improvement in retrieval accuracy [2]. However, this work highlighted a number of opportunities for improving the way in which enrichment queries are generated, and the way in which enrichment results are selected prior to enrichment term extraction and in this paper we propose improvements in both of these areas and demonstrate additional performance gains.

2 Related Work

The research area we have identified as most relevant to our current work is the *query expansion* or *relevant feedback* domain. A well known issue in Web search and IR is that short queries and term mismatching can cause relevant documents to be omitted from search results because they do not contain the terms within a user's query. The idea behind query expansion is that if we *enrich* the query using terms from a set of relevant documents, overall retrieval performance increases. One of the most popular query expansion techniques is known as *relevance feedback* [6, 7]. The standard approach involves a user submitting an initial query to the system, receiving a set of results and indicating which results

are relevant. Terms are then extracted from these *relevant* documents and are used to supplement the users initial query. This *iterative* process continues until the users information need is satisfied.

Although existing research shows that relevance feedback can achieve significant improvements in retrieval performance, it requires that users provide accurate relevance judgments. However, users are often reluctant to provide such information. To overcome this difficulty, the concept of *pseudo-relevance* or *blind feedback* was introduced. In this approach the system handles relevance feedback by assuming that the top-ranked documents returned to a given query are relevant. Terms are then extracted from these documents and are used to formulate a new enriched query. Previous studies have shown that pseudo-relevance can lead to significant improvements in retrieval performance [1, 3]. Our approach to content enrichment is similar in spirit to *pseudo-relevance feedback* domain except that top-ranking documents are used as a source of indexing terms (at indexing time) as opposed to a source of query terms at search time.

3 Mobile Content Enrichment

The main contribution of this paper is to describe a technique for enriching mobile content by leveraging existing Web search resources. The objective is to expand the limited content of a typical mobile page to produce an enriched version of this page for the purpose of indexing. The enrichment process involves using elements of the page's original content as queries to a Web search engine, with the *enrichment terms* extracted from the top ranking results retrieved for these queries. The assumption is that this will lead to enrichment terms that are missing from the original content (document) but that are nevertheless useful for indexing, and that thus provide for additional retrieval opportunities.

Previously we [2] described a basic enrichment technique, focusing on how enrichment terms are extracted from the search engine results, and demonstrating search performance improvements in excess of 20%. In this paper, we will extend this work by focusing on two different stages of the enrichment process— (1) the extraction of query terms from the original content; and (2) the selection of results as a source of enrichment terms — and will demonstrate how these improvements have a further impact on search engine performance.

3.1 The Enrichment Process

To provide a general overview of the basic enrichment process, consider a mobile page *S'*. Enrichment is then a 5-stage process:

1. *Query Extraction*: Generate a query, $Q(S')$, or rather a set of q queries from S' by extracting k informative terms.
2. *Result Extraction*: Submit each query $Q_i(S')$ to a Web search engine (by default we use Yahoo) to generate a set of r results, $R_{Q_i(S')}$.
3. *Result Selection*: Select the r highest quality results from $R_{Q_i(S')}$ to produce a filtered set of Web search results, $R^{filtered}_{Q_i(S')}$.

4. *Term Extraction*: Analyse the content of these filtered results to extract a set of the n most informative enrichment terms $V(S') = t_1, ..., t_n$.
5. *Page Indexing*: Index S' using a combination of its own terms and the enrichment terms; that is, $E(S') = S' \cup V_{Q(S')}$.

In this way each mobile page S' is expanded by a set of enrichment terms which have been selected because they appear to be related to the content of S'. The process above can be tuned in a variety of ways, by varying key parameters (such as the number of enrichment terms to add or the number of related results to select as a source of these terms) to facilitate a narrow or broad approach to enrichment. If the enrichment terms are too narrow then new retrieval opportunities may be limited. If they are too broad, while new retrieval opportunities may be readily available, retrieval precision may be reduced.

As mentioned above, our previous work [2] has focused on *Step 4* of the enrichment process, while using very straightforward approaches in the other steps. In the following sections we will focus on *Steps 1 & 3*. The assumption is that by improving the way in which we generate queries from the source content we can more reliably identify result pages that are likely to provide a good source of enrichment terms. Similarly, by being more selective in the result pages that we use as a source of these terms, we can produce a better set of final enrichment terms; for example, eliminating pages that are among the top retrieved results but that do not appear to be relevant to S' will help to improve the final quality of the enrichment terms.

3.2 Query Generation

The default approach to generating a query from S', as reported in [2], was based on generating a single query for each S' from the top k most frequently occurring terms in S' (after stop-word removal). While this approach appeared to work reasonably well it is clear that there is considerable room for improvement. As such we propose two new strategies that involve the generation of multiple queries, instead of a single query, using two different query extraction techniques:

1. *MTF (multiple queries, term frequency extraction)*: Instead of generating a single query, we generate q ($q = 10$) queries of size k ($k = 5$) from the terms with the highest frequency in S'. Specifically we extract the top 10 terms with the highest frequency in S' and generate a list of q unique 5-term query combinations from these top 10 terms.
2. *MYH (multiple queries, Yahoo term extraction)*: Generate a set of q ($q = 10$) queries by submitting S' to the *Yahoo Term Extraction Tool* [4]. This tool returns a ranked list of key terms from a body of text (S'), which we use as a source to extract q $k - term$ queries as above (with $q = 10$ and $k = 5$).

[4] http://developer.yahoo.com/search/content/V1/termExtraction.html

3.3 Result Selection

In the enrichment technique employed by Church and Smyth [2], the results chosen as a source of enrichment terms were simply all of the top-ranking results returned by the underlying search engine for some query $Q_i(S')$. The problem with this approach is that it can lead to the inclusion of a result that has little or no relevance to the source content (S'), especially in the face of a vague query.

What is needed is a technique for filtering out result pages that are unlikely to serve as a good source of enrichment terms. One way to do this is to compare each result r_j to S'. If there is a significant overlap between the result and S' then we can reasonably infer a level of similarity or relatedness. Overlap on its own does not go far enough however because we would like to favour results that are related but that also offer extra terms that are missing from S'; after all r_j may be very related to S', but if $S' - r_j$ is essentially empty then r_j will not act as a source of new enrichment terms. For this reason we evaluate each result r_j by considering its overlap with S' and the availability of new terms that are missing from S'. We prefer results that have a significant overlap *and* that offer a significant number of new terms by using a harmonic mean of both factors to evaluate result quality; see Equations 1, 2, and 3. Thus, in this paper we will examine a result selection approach (Sel) which selects the top m Yahoo results with the highest quality scores. In this way high-ranking results that fall below this threshold will not be considered as a source of enrichment terms.

$$Quality(r_j, S') = \frac{1}{\frac{1}{Overlap(r_j,S')} + \frac{1}{Diff(r_j,S')}} \quad (1)$$

$$Overlap(r_j, S') = \frac{|r_j \cap S'|}{|S'|} \quad (2) \qquad Diff(r_j, S') = \frac{|r_j - S'|}{|r_j|} \quad (3)$$

4 Evaluation

We have argued that enrichment is one way to improve the retrievability of mobile content and that existing search engines can be a valuable source of enrichment knowledge. In this section we will evaluate the impact of our new query generation and result selection components compared to a standard mobile search engine and the basic approach to enrichment described by [2].

4.1 Test Data

In this experiment we use a database of 1999 recent news stories, harvested from a Web-based news service during October 2005 - February 2006, as the basis for a mobile news search engine. News stories were chosen for two important reasons. First of all, news is a good example of the type of content that is popular on the mobile Internet. Second, it is relatively easy, for the purpose of this experiment, to convert a long-version of a news story into a shorter, mobile version that will form the basis of the mobile news content. Typically these shorter versions

were about 10-20% of the original and, in the case of our news content, this truncation process was straightforward because each story was preceded by a concise summary of the longer text. The point of this is that the additional story content, which did not make it into the mobile form of the story, was then available as a plausible source of potential target test queries. In this way, during performance testing, for each indexed mobile story (*target story*), we generated test queries (using the same technique for generating enrichment queries) from the content that was missing from the mobile form; thus test queries will often contain terms that are not present in the mobile version of the story, but that are nonetheless relevant as potential query terms. These test queries were submitted to each test search engine and we measured the percentage of times that the target story was retrieved among the top 10 results.

4.2 Test Search Engines

In all we evaluated 6 different test search engines, each corresponding to a different approach to indexing and enrichment: $SE1$ used no enrichment and news stories were indexed using their existing content only; the remaining 5 search engines implement different version of our enrichment strategies. All of the search engines were implemented using the *Lucene*[5] platform and Yahoo was used as the underlying search engine responsible for providing enrichment results. The summary details for each search engine are as follows:

1. *SE1*: This search engine provided a baseline as it used only standard mobile content to index the news stories.
2. *TF*: This search engine uses a basic enrichment technique (described by [2]) in which each news story is enriched from the top 10 ranking results returned from Yahoo based on a single enrichment query made up of the top k most frequent terms in the mobile story.
3. *MFT*: This search engine uses the same enrichment technique as TF except that 10 enrichment queries are used to provide enrichment results.
4. *MYH*: This search engine uses the same enrichment approach as MTF except that the enrichment queries are generated using the Yahoo Term Extraction Tool as discussed in Section 3.2.
5. *MTFSel*: This variation operates in the same way as MTF except that in addition, the enrichment results are filtered using the quality metric discussed in Section 3.3.
6. *MYHSel*: This variation operates in the same way as $MTFSel$ except that the enrichment queries are generated using the Yahoo Term Extraction Tool as discussed in Section 3.2 and results are filtered using the quality metric as discussed in Section 3.3.

Note that k, the number of terms used in our mobile query $Q(S')$, is set to 5 for each search engine in our experiments.

[5] http://lucene.apache.org

4.3 Methodology

An experimental run involves testing the retrievability of each of the 1999 news stories. For each *test story*, we generate a set of test queries q, where $q = 3$ from its non-mobile content. To generate the test queries we vary p (the number of terms used in test queries) between 1,2 or 3 terms by selecting the t most frequent terms in the remaining portion of news content that is not part of the original news stories S'. Next we submit each test query to the 6 test search engines and compute the average percentage accuracy (we will refer to this as the *success rate*) for each of the test search engines as the percentage of times that the target result (the current test story) is found in the top 10 results returned. This is repeated for a variety of different parameterizations of our content enrichment technique by varying $r = 3, 10$ (the number of Yahoo results extracted) and $n = 10, 50, 100, 200$ (the number of terms used for enrichment), to understand how retrieval performance varied under different experimental conditions.

4.4 Overall Successful Rates

Figure 1 presents the overall success rate for each of the 6 test search engines averaged over all parameter settings. Although the average success rates are reasonably low—e.g., the results show that indexing mobile news stories by their own content delivers successful retrievals approx. 20% of the time, on average—we must remember that the procedure we are using enforces a strict notion of relevance, in the sense that only one news story is considered to be relevant for each query. In reality, searchers are likely to entertain a variety of relevant results for their queries. As such it is reasonable to interpret the results as lower bounds on search engine performance. The average searcher is likely to enjoy better search engine performance in practice, but we predict that comparable relative differences between the search engines will remain under such weaker relevance conditions.

Returning to the success rates for each of our test search engines, we find that the basic search engine $SE1$ (which uses no enrichment), fails to find the target news story in 79% of cases. In contrast, we see that on average, all of the enriched engines do significantly better. For example, the basic enrichment technique (TF) offers improvements of around 23% over $SE1$. However, the results also show that each of our enhancements to the basic enrichment technique have a positive effect on retrieval performance. For example, MTF and MYH take advantage of multiple enrichment queries to offer improvements of 27% over $SE1$. $MTFSel$ which incorporates a quality-based result selection phase offers an even greater overall benefit (34%). Overall, the best performing technique, $MYHSel$, combines the use of multiple queries, the more sophisticated Yahoo term extraction service to generate enrichment queries, and quality-based result selection, to achieve a relative success rate increase of 37% over SE1.

These results clearly demonstrate that we can significantly increase retrieval performance by (1) using multiple queries at the query generation phase, (2) using more sophisticated term extraction techniques, and (3) employing the quality metric to filter out any *low-quality* enrichment results.

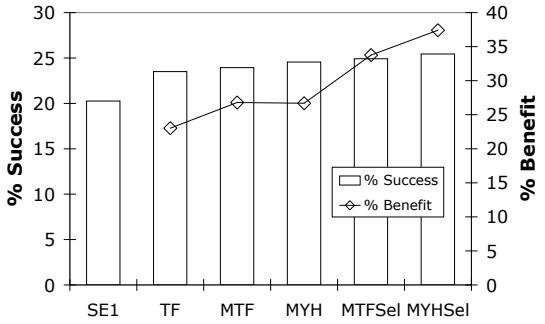


Fig. 1. Average overall percentage success rates for each of the 6 search engines and the relative benefit of the 5 enriched search engines compared to SE1.

4.5 Varying n: the number of enrichment terms

Varying the number of additional enrichment terms is likely to have a significant impact on retrieval performance. The more relevant the terms selected for indexing, the better able the search engine will be to return the target document for a relevant query. The risk, however, is that if too many irrelevant (or at least less relevant) terms are used to index a document then this will increase the likelihood of false remindings at search time. If this happens we can expect to find many irrelevant documents being retrieved during a typical search, thereby increasing the likelihood that the target document will be pushed further down the result-list and thus reduce search success.

The results obtained by varying n, the number of enrichment terms used during indexing, are presented in Figure 2 and 3, by averaging the experimental runs for each particular value of n. Once again, we find a significant advantage accruing to the enriched search engines. Indeed we see that as the enrichment terms are increased from 10 to 200 there is a steady improvement in overall search engine performance, relative to $SE1$. At only 10 enrichment terms there is little significant advantage for the enriched search engines but the use of 200 enrichment terms sees a minimum relative improvement of 33% for the basic enrichment technique TF, while $MYHSel$ achieves a 73% relative improvement over $SE1$; note 73% is the average of the individual relative benefits found for $MYHSel$ during the $r = 3$ and $r = 10$ runs, at $n = 200$.

Again it is clear that the use of multiple queries, the more sophisticated term extraction technique for query generation, and the quality filter for result selection, all have an independently positive effect on overall retrieval performance. For example, the use of multiple queries contributes up to 20% in relative benefit terms, whereas the use of the more sophisticated term extraction technique adds another 6%, and the result selection filter adds a further 14%.

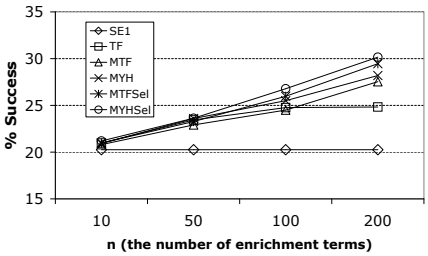


Fig. 2. Search success rates

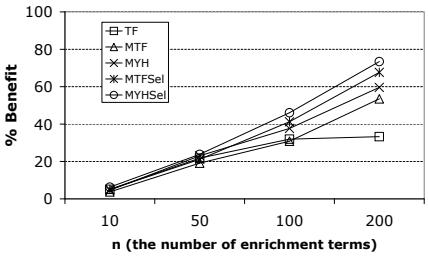


Fig. 3. Average benefit relative to SE1

4.6 Varying r: the number of enrichment results

Another critical parameter in the enrichment process is the number of results that are retrieved in response to each enrichment query. These results provide the raw material from which the enrichment terms are extracted. If too many results are retrieved then it is likely that less relevant sources of enrichment terms will be considered during the term selection phase. This will potentially reduce the quality of enrichment terms. At the same time, if we retrieve too few results then we are limiting the enrichment process to a reduced set of potential enrichment terms and this could lead to missed indexing opportunities.

The results for 2 different settings for r (3 and 10) are presented in Figure 4 and 5. As before we see that the enriched search engines all do significantly better than $SE1$ across the different values for r. Interestingly, using fewer results ($r = 3$) is seen to have a positive impact on overall retrieval performance. For $r = 10$ each of our enriched search engines offers improvements over $SE1$ of between 23%-33%. However, when $r = 3$ we see improvements in the range 23%-42%. The results suggest that higher ranking results are a better source of enrichment material and while the result-selection filter helps to control the quality of the results used in enrichment, at $r = 10$ some lower quality results are being incorporated. Remember that in this experiment we extract the top r results with the highest threshold. At $r = 3$ the average quality of selected results is approximately 0.4, but at $r = 10$ this average quality decreased to 0.36. Thus the higher quality results selected for $r = 3$ have contributed to an overall increase in performance for this setting.

5 Conclusions

Mobile search is challenging because of fundamental device and content limitations. Mobile content, by its very nature, tends to be brief and can lack the richness that is needed to build a good search index and, in this paper, we have addressed this problem by presenting a heuristic content enrichment framework that leverages traditional Web search engines and tools as a way to enrich mobile content prior to indexing. We have described a number of enhancements to

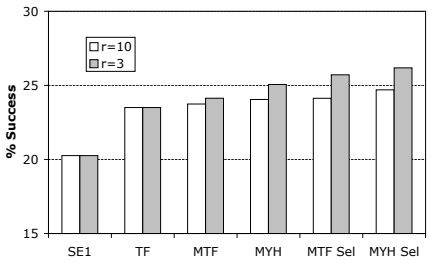


Fig. 4. Search success rates

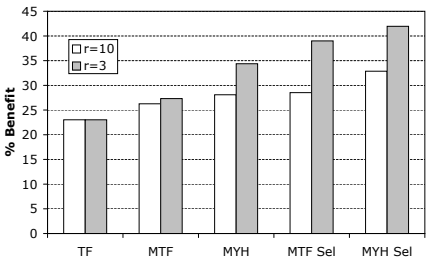


Fig. 5. Average benefit relative to SE1

our basic content enrichment strategy and evaluated the performance of each of these enrichment strategies, relative to a benchmark mobile search engine, and under a variety of experimental conditions. The results demonstrate significant improvements in overall search engine performance.

This content enrichment framework provides many opportunities for further developments beyond the term-based selection and weighting techniques used to date. For example, a logical next step is to take advantage of linguistic knowledge and simple natural language processing techniques in order to guide more meaningful enrichment. Moreover, there are a number of opportunities to look at the application of machine learning techniques to the extraction of suitable enrichment terms and the classification of indexing knowledge, for example.

References

1. C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic Query Expansion Using SMART: TREC 3. In *Proceedings of the TREC-3*, 1994.
2. K. Church and B. Smyth. Mobile Content Enrichment. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI'07)*, pages 112–121, 2007.
3. E. N. Efthimiadis and P. V. Biron. UCLA-Okapi at TREC-2: Query Expansion Experiments. In *Proceedings of TREC-2*, pages 278–290, 1993.
4. G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The Vocabulary Problem in Human-System Communication. *Communications of the ACM*, 30(11):964–971, 1987.
5. M. Ramsay and J. Nielsen. The WAP Usability Report. Neilson Norman Group. 1983.
6. J. J. Rocchio. Relevance Feedback in Information Retrieval. *Gerard Salton, editor, The SMART Retrieval System - Experiments in Automatic Document Processing*, pages 313–323, 1971.
7. G. Salton and M. McGill. Introduction to Modern Information Retrieval. 1983.

An Axiomatic Comparison of Learned Term-weighting Schemes in Information Retrieval

Ronan Cummins and Colm O'Riordan

Dept. of Information Technology,
National University of Ireland,
Galway, Ireland.
ronan.cummins@nuigalway.ie, colm.oriordan@nuigalway.ie

Abstract. Learning approaches in information retrieval are becoming increasingly widespread. One learning paradigm that has attracted some attention recently has been genetic programming. In this paper, we present term-weighting functions reported in the literature which were developed by four separate approaches using genetic programming.
Recently, three axioms (constraints), from which all *good* term-weighting scheme should be deduced, have been developed and shown to be theoretically and empirically sound. We introduce a new axiom which follows intuitively from the original three. We analyse the $BM25$ scheme and the four learned schemes presented to determine if the schemes are consistent with all four axioms. We find that only one term-weighting approach is consistent with all the axioms. Finally, we empirically evaluate all of the schemes on a number of test collections for short, medium and long TREC queries and subsequently show that the only scheme that is consistent with all of the axioms outperforms the other schemes.

1 Introduction

Term-weighting is crucial to the problem of document ranking within most, if not all, information retrieval (IR) systems. Many approaches to term-weighting have been developed over the years. The derived schemes have been produced from various types of models, ranging from empirical learning models to purely theoretical models. With the increase in computing resources and the advances in machine learning techniques, many attempts have been made to learn term-weighting schemes assuming a *bag of words* approach. In particular genetic programming (GP) is becoming popular due to the freedom offered in the definition of the problem and representation of the possible solutions. The basis behind many of these approaches is that useful features and properties within a population of solutions survive and propagate. GP produces a symbolic representation of the solution, which is useful for analysis and generalisation. There have been a number of attempts using GP to evolve term-weighting schemes for ad hoc retrieval [7, 4, 10, 2].

An axiomatic approach to IR [5] has been employed which develops a number of constraints (axioms) to which all *good* weighting functions should adhere. This approach and the constraints in particular, are useful in attempting to theoretically motivate term-weighting functions that are developed from purely automated learning (empirically based) models.

More precisely, we believe that the best functions produced from a valid learning approach to IR should adhere to these existing constraints (axioms). The satisfaction of the constraints serve as a useful guide to the optimality of the solutions produced. These constraints can potentially be used in a number of different ways. They can be used to validate a specific learning paradigm in an IR domain by showing that the solutions (term-weighting schemes) produced adhere to them (i.e. that the learning approach adopted navigates the search space effectively and finds the area of the space in which good term-weighting schemes lie). The axioms can also be used to constrain the search space so that the learning approach is searching a much smaller space. In the latter case, the learning approach can search a space in which good solutions are already known to exist.

This paper presents an analysis, using the existing axiomatic framework for IR, of term-weighting schemes learned using four different GP approaches. Firstly, we introduce the three existing axioms and postulate a new axiom which complements the original axioms. We present one benchmark term-weighting scheme ($BM25$) and four learned term-weighting schemes from separate GP approaches to the problem. All of the learned term-weighting schemes herein have been developed using a GP process in a *bag of words* retrieval framework. Furthermore, we present results which show that the only scheme which obeys with all the axioms outperforms *all* others over multiple collections for various types of query without the need for tuning parameters.

The remainder of the paper is organised as follows: Section 2 introduces the existing axioms and motivates a fourth axiom. Section 3 presents five term-weighting approaches together with a brief analysis of each scheme. In section 4, we present results which provide empirical evidence for the validity of the four axioms. Finally, our conclusions are outlined in section 5.

2 Axioms for Term-Weighting

We will briefly introduce previously developed constraints using an inductive framework [5]. The idea of this inductive framework is to define a base case function that describes the score (weight) assigned to a document containing a single term matching (or not matching) a query containing a single term. All other cases can be dealt with inductively using two separate functions. A *document growth function* describes the change in the document score when a single term is added to the document, while a *query growth function* describes the change in the document score when a single term is added to the query. This is an elegant approach to formalising necessary characteristics of *good* term-weighting functions.

Three axioms have been postulated and these can be used to validate or to develop term-weighting schemes in a constrained space. Thus, we used the term axiom and constraint analogously in this paper. The first constraint (constraint 1) states that adding a new query term to the document must *always* increase the score of that document. This captures the basic behaviour of a term-frequency aspect. The second constraint (constraint 2) states that adding a non-query term to a document must *always* decrease the score of that document. This constraint ensures that some sort of normalisation is present and specifies its basic operating principle. The third constraint (constraint 3) states that adding successive query terms to a document should increase the score of the document less with each successive addition. Essentially, the term-frequency influence must be sub-linear. The intuition behind this constraint is that it is ultimately the first occurrence of a term that indicates that the document is on-topic (i.e. related to the query). Due to characteristics of natural language, it is known that when a term first appears, when reading through a document, the likelihood of re-appearance increases. Thus, the weight given to successive occurrences of a query term should be reduced.

These constraints are used to check the validity of term-weighting schemes before evaluation. Furthermore, term-weighting schemes which adhere to these constraints are shown empirically to outperform weighting schemes that fail to adhere to one or more of the constraints [5]. The constraints are also useful in defining valid bounds on tuning parameters that appear in many existing term-weighting schemes. It should be noted that simply adhering to these constraints does not guarantee a high performance weighting scheme. Rather it is the violation of one or more of the constraint that indicates the performance is non-optimal (i.e. breaks some rule of the proposed model of relevance). It is worth noting that these axioms *typically* constrain a term-weighting schemes within-document features (i.e. its term-frequency aspect and normalisation aspect).

2.1 New axiom

We now propose a new constraint which aims to avoid over-penalising successive occurrence of non-query terms in documents. $F(Q, D)$ is a function which scores a document D in relation to a query Q in a standard *bag of words* retrieval model. With notation similar in style to [5], our new constraint can be formalised as follows, where $t \in T$ is a term t in the set of terms in a corpus and $\delta_t(t, D, Q) = F(Q, D \cup \{t\}) - F(Q, D)$ (i.e. the change in score as t is added to the document D):

Constraint 4: $\forall Q, D$ and $t \in T$, if $t \notin Q$, $|\delta_t(t, D, Q)|^{-1} > |\delta_t(t, D \cup \{t\}, Q)|^{-1}$. According to Heaps' law [6], the appearance of new unseen terms in a corpus grows in roughly a square-root relationship (sub-linearly) to the document length (in words). Therefore, as *non-query terms* appear in a document they should be penalised less with successive occurrences. This constraint avoids over-penalising longer documents by ensuring that the normalisation aspect is sub-linear.

Essentially, the inverse of the score reduction due to non-query terms being added (an increasing value) should be sub-linear. This follows intuitively from constraint 3 which controls how the score of a document changes as successive query terms are added to a document. As such, it is the first appearance of a non-query term that ultimately indicates a change in the topic of a document and successive occurrences of this term do not indicate that the topic of that document is drifting from the query to the same degree.

3 Term-weighting Analysis

This section presents the $BM25$ benchmark function and four learned functions found in the literature. Each function is briefly analysed using the axioms previously introduced. Table 1 summarises the conclusions from this section.

3.1 *BM25*

The $BM25$ weighting scheme, developed by Robertson et al. [9], is a weighting scheme based on the probabilistic model. The score of a document D in relation to a given query Q can be calculated as follows:

$$BM25(D,Q) = \sum_{t \in Q \cap D} \left(\frac{tf_t^D}{tf_t^D + k_1 \cdot ((1-b) + b \cdot \frac{dl}{dl_{avg}})} \cdot log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot tf_t^Q\right) \tag{1}$$

where tf_t^D is the frequency of a term t in D and tf_t^Q is the frequency of the term in the query Q. dl and dl_{avg} are the length and average length of the documents respectively measured in non-unique terms. N is the number of documents in the collection and df_t is the number of documents in which term t appears. k_1 is the term-frequency influence parameter which is set to 1.2 by default. The query term weighting used here (tf_t^Q) is slightly different to the original weighting method proposed [9] but has been used successfully in many studies [5]. b is the document normalisation influence parameter and has a default value of 0.75.

Analysis. It can be noted that the idf component in the $BM25$ ($log(\frac{N-df_t+0.5}{df_t+0.5})$) function will return a negative value when $df_t > \frac{N}{2}$ and thus violates constraint 1 and 3 in certain circumstances. However, this typically violates these constraints when stop-word removal is not used, as very frequent terms would otherwise be removed [5]. It can also be seen that the normalisation function used in this function is linear. This suggests that it needs to be tuned on specific collections as it may over-penalise long documents on certain collections.

3.2 Oren

One of the first approaches to evolve term-weighting schemes using GP was conducted in [7, 8]. This approach used non-atomic features of the terms, documents, queries and the collection to evolve term-weighting functions for use

in IR. Using parts of existing functions as terminals in the GP can be viewed as a type of seeding or biasing, as prior knowledge about what constitutes a good ranking function is assumed. This type of approach can arbitrarily limit the search space. This work uses a small document collection (1,239 documents) and 70 queries to evolve functions using a population of 100 individuals run for 150 generations. It has been noted by the author that they believe that many of their functions are non-generalisable. One of the schemes outlined in this work [8] can be re-written as follows:

$$F1(D,Q) = \sum_{t \in Q \cap D} \left(\frac{tf_t^D}{tf_t^D + df_t + dl \cdot (1 + 0.436 \cdot \frac{tf_t^D}{tf_{max}^D} \cdot (cf_{max} + log(cf_{max})))}\right) \tag{2}$$

where tf_{max}^D is the frequency of the most common term in D, cf_{max} is the frequency of the most common term in the collection.

Analysis. This function can be written as $\frac{tf_t^D}{tf_t^D + df_t + dl \cdot (1 + tf_t^D \cdot K_1)}$ where K_1 is a constant such that $K_1 > 0$, which can be rewritten as $\frac{tf_t^D}{tf_t^D \cdot (1 + dl \cdot K_1) + df_t + dl}$. In this form it is easier to see that the equation satisfies constraints one and three. An extra occurrence of a query term will always increase the weight of a document and this increase in weight is sub-linear. Constraint two is also satisfied as the weight of a document will always decrease as non-query terms are added. However, it can be determined that the normalisation component is linear in nature (i.e. the normalisation component dl is linear in the denominator). This is probably due to the fact that it was learned on one collection.

3.3 Fan et al.

Another approach to evolving weighting schemes [4] which assumes a simplistic query term weighting (i.e. tf_t^Q) has also been attempted. In this research, a term-weighting function was learned using short queries and a population of 200 individuals for 30 generations. The best functions outlined in [4] can be re-written as follows:

$$F2(D,Q) = \sum_{t \in Q \cap D} \left(\frac{log(tf_t^D \cdot X)}{vl + 2 \cdot tf_{max}^D + 0.373}\right) \cdot tf_t^Q \tag{3}$$

where

$$X = \left(tf_{avg}^D + \frac{tf_t^D}{log(tf_t^D \cdot 2 \cdot tf_{avg}^D)} + \frac{tf_t^D \cdot N \cdot tf_{avg}^D \cdot (tf_{max}^D + vl)}{df_t^2}\right) \tag{4}$$

where tf_{avg}^D is the average term-frequency in D and vl is the length of the document vector (unique terms).

Analysis. In this scheme, a new occurrence of a query term always increases the weight of a document and this increase in weight is sub-linear. As a result constraint 1 and 3 are satisfied. However, as the normalisation used is the number of unique terms (vector length), the second and fourth constraints are violated. Consider a non-query term, which has already appeared in the document. If this term re-occurs, the weight of the document will not decrease as the vector length remains unchanged. Even if the document length factor used was changed to the document length including repetitions (i.e. dl), constraint 4 would still be violated.

3.4 Trotman

An approach using primitive atomic features of terms, documents, queries and the document collection has also been attempted [10]. This approach used a large terminal and function set with little or no constraints on the search space. In this approach a seeded population of 100 individuals (96 random and 4 existing functions) was ran for 100 generations 13 times. One of the best performing schemes (named run 5 in [10]) can be re-written as follows:

$$F3(D,Q) = \sum_{t \in Q \cap D} (log_2 |\frac{N - log_2|N|}{2 \cdot df_t}| \cdot \frac{cf_t}{df_t} \cdot \frac{tf_t^D \cdot tf_t^Q \cdot cf_{max}}{max(C_3, C_4 + \frac{C_1 \cdot (log|C_2 + tf_t^Q| + cf_t) \cdot dl}{N \cdot dl_{avg}}) + tf_t^D}) \quad (5)$$

where cf_t is the frequency of t in the entire collection of N documents. C_1, C_2, C_3 and C_4 are constants of value 33.40102, 23.94623, 1.2 and 0.25 respectively.

Analysis. Firstly, the $log_2|\frac{N-log_2|N|}{2 \cdot df_t}|$ part can lead to a negative weight for terms with a high document frequency. This leads to constraint 1 and 3 being violated in circumstances similar to the $BM25$ scheme. Also, we can see that when $C_3 > C_4 + \frac{C_1 \cdot (log|C_2 + tf_t^Q| + cf_t) \cdot dl}{N \cdot dl_{avg}}$, which typically occurs when cf_t is low (i.e. for rare terms), the within document part of the formula reduces to $\frac{tf_t^D}{1.2 + tf_t^D}$ which is a primitive form of the $BM25$ local weighting. This form of the function has no normalisation component and thus violates the second constraint. However, this form conditionally satisfies the first and third constraints as adding a query term to a document always increases its score (constraint 1) and is sub-linear (constraint 3).

When $C_3 < C_4 + \frac{C_1 \cdot (log|C_2 + tf_t^Q| + cf_t) \cdot dl}{N \cdot dl_{avg}}$, which typically occurs when the collection frequency for a term is high (i.e. more common terms) a different form of the function is used. Consider a typical case when tf_t^Q is 1 and N is large (e.g. 100,000). In such a case, this reduces to approximately $0.25 + \frac{3.2 + cf_t}{3000} \cdot \frac{dl}{dl_{avg}}$. For high values of cf_t, this will exceed C_3 (i.e. 1.2). Interestingly, this can be re-written as $\frac{tf_t^D}{0.25 + K_2 \cdot \frac{dl}{dl_{avg}} + tf_t^D}$ (where K_2 is a global constant for a particular term) which contains a normalisation form similar to the $BM25$ scheme. When the

function takes this form all three original constraints [5] are satisfied. However, the normalisation scheme is not sub-linear and thus does not conform to the new constraint (constraint 4). An interesting aspect to note is the evolution of a density measure (i.e. $\frac{cf_t}{df_t}$) which has been independently evolved in other work [2].

3.5 Cummins-O'Riordan

In this approach, a three-stage incremental approach was used to develop an entire term-weighting scheme by evolving, in turn, three constituent parts of a function similarly to [3]. The term-discrimination (or global) part was developed using a population of 100 for 50 generations, while the term-frequency and normalisation parts were developed using a population of 200 for 25 generations each. The following is a typical entire term-weighting scheme:

$$F4(D,Q) = \sum_{t \in Q \cap D} (\frac{ntf}{ntf + 0.45} \cdot \sqrt{\frac{cf_t^3 \cdot N}{df_t^4}} \cdot tf_t^Q) \tag{6}$$

The term-frequency influence factor here (i.e. $\frac{ntf}{ntf+0.45}$) has been modelled to reflect the effect of an evolved term-frequency influence function by measuring the relative term-frequency as in [1]. The normalisation aspect was evolved on three different collections (indexed seperately) with varying document length characteristics as most normalisation functions (including $BM25$) tend to be collection specific. One of the best normalisation functions found using this approach was $\sqrt{\frac{dl}{dl_{avg}}}$.

$$ntf = \frac{tf_t^D}{\sqrt{\frac{dl}{dl_{avg}}}} \tag{7}$$

Analysis. It can be seen that the term-discrimination part of this function is always positive and the term-frequency is sub-linear. The normalised term-frequency (ntf) is normalised as $tf_t^D / \sqrt{\frac{dl}{dl_{avg}}}$. Thus, if a query term is added both tf_t^D and dl will increase by 1, thereby increasing the value of the ntf component. As a result, constraint 1 and constraint 3 are satisfied. If a non-query term is added, ntf will decrease in value as dl will increase by 1; this satisfies constraint 2. This normalisation factor is also sub-linear, which leads to constraint 4 being satisfied.

3.6 Summary of Constraint Satisfaction

Table 1 shows the constraints that each scheme satisfies. The conditional satisfaction (denoted Cond.) means that the constraint is satisfied in many circumstance (as previously noted) but does not unconditionally satisfy the constraint. We have ranked these schemes based on how many constraints they satisfy. We

ranked $BM25$ and $F3$ ahead of $F2$ mainly due to the fact $BM25$ and $F3$ will only violate constraint 1 and constraint 3 if stop-words are not removed which is not typically the case in many IR systems. Thus, $F2$ will typically break constraints more often than $BM25$ or $F3$ will.

Table 1. Constraints Satisfaction

		Constraints			
Rank	Scheme	One	Two	Three	Four
1	$F4(D,Q)$	Yes	Yes	Yes	Yes
2	$F1(D,Q)$	Yes	Yes	Yes	No
3	$BM25(D,Q)$	Cond.	Yes	Cond.	No
4	$F3(D,Q)$	Cond.	Cond.	Cond.	No
5	$F2(D,Q)$	Yes	No	Yes	No

It should be noted that this ranking is coarse as we do not know if violations of different constraints lead to equal levels of suboptimality. We are also unsure if the schemes identified are specific to a type of query or indeed the specific environment in which they were trained. Nonetheless, given these details of constraint satisfaction it seems an intuitive and possibly useful way of ordering the schemes by expected performance.

4 Empirical Comparison

In this section we present experiments to empirically validate the previous analysis. We tested these five schemes on different types of collections and queries (topics) to measure the MAP (mean average precision) of each scheme. The use of stop-word removal in our experiment leads to constraints 1 and 3 being satisfied for $BM25$ and $F3$ on the collections used herein. This has been determined empirically.

4.1 Document Collections

We used the LATIMES and FT (years 1991 to 1993) collections from TREC disks 4 and 5 as test collections and topics 401 to 450. For each set of topics we create a short query set, consisting of the title field of the topics, a medium length query set, consisting of the title and description fields, and a long query set consisting of the title, description and narrative fields. We also use documents from the OHSUMED collection (years 1990 and 1991) as a test collection. We created short queries for the OHSUMED collection by simply removing terms from the description field of the 63 topics. Standard stop-words from the Brown Corpus[1] are removed and remaining words are stemmed using Porter's algorithm. We also used the NPL[2] collection as a smaller test collection on which to judge the schemes.

[1] http://www.lextek.com/manuals/onix/stopwords1.html
[2] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/

4.2 Experimental Results

Table 2. %MAP on four different collections

	FT91-93 (401-450)			OHSUMED (63)		NPL (93)	LATIMES (401-450)		
#docs	(138668)			(148162)		(11429)	(131896)		
Schemes	short	medium	long	short	medium	medium	short	medium	long
$F4$	32.72	36.30	38.46	25.53	30.07	28.95	26.50	26.87	28.80
$F1$	23.13	23.40	18.92	17.70	13.51	25.76	15.40	13.73	11.83
$BM25$	31.27	35.33	35.35	25.54	28.08	28.74	24.85	26.73	28.86
$F3$	31.95	33.82	33.61	23.19	26.27	27.54	24.30	25.81	25.68
$F2$	26.61	15.40	08.00	06.90	01.23	02.80	12.56	08.74	02.59

The results (Table 2) show that $F4$ outperforms most of the other schemes on the various test data. This provides evidence to validate our new constraint. The remaining schemes tend to perform in accordance with the rank in Table 1. However, $F1$ is the exception to this rule. This scheme adheres to three of the constraints but performs poorly on many of the collections. This scheme was learned on a very small set of documents (1,239 documents) and it would appear that this is not a suitable size collection to be generalisable. Indeed, it is suggested by one of the scheme's authors in [7] that this formula is not likely to be generalisable to any degree.

4.3 Discussion

The existing axioms and the newly postulated axiom are useful estimators of term-weighting optimality. As such, they can be useful in estimating the performance of a scheme. An interesting result is that many of the learned approaches conditionally adhere to some of the constraints. This would suggest that they did indeed learn useful methods for weighting terms in their training environment but that their training data is quite specific (i.e. the constraints are satisfied for the characteristics on the training data but are not unconditionally satisfied).

These results can tell us something about how to learn term-weighting functions. Small collections (less than 10,000 documents) should be avoided when aiming to learn *generalisable* term-weighting schemes. Indeed, it has already been shown in [2] that the term-discrimation (global) part of a term-weighting scheme can indeed be learned on a small collection but it is typically the within-document (local) part of these schemes that is not generalisable. Although the solution learned from such an approach ($F1$) does indeed adhere to the three original axioms it is not generalisable.

To overcome the collection dependance problem (which typically affects the type of normalisation to use), it is advisable to use multiple varied training collections indexed seperately in order to learn schemes that will adhere to the constraint specified herein. On a related note, it can be determined that using

the vector length as a measure for document length will lead to two constraint violations as outlined in section 3.3.

Furthermore, it is advisable to use medium or long queries when learning such term-weighting schemes. Short queries (as used in [4]) do not provide as much information about how terms should interact with each other (particularly in a term-discrimination context). This will lead to an increase in training time.

5 Conclusions

A new axiom for information retrieval has been introduced. We have presented four learned term-weighting schemes and one term-weighting scheme which was analytically developed. Only one of the schemes is consistent with all four axioms. Interestingly, this scheme is one of the learned schemes. An evaluation of the term-weighting schemes validates the analysis.

An interesting future direction would be to constrain the search space using the axioms and then use a learning technique to search this reduced space.

References

1. Ronan Cummins and Colm O'Riordan. An evaluation of evolved term-weighting schemes in information retrieval. In *CIKM*, pages 305–306, 2005.
2. Ronan Cummins and Colm O'Riordan. Evolving local and global weighting schemes in information retrieval. *Information Retrieval*, 9(3):311–330, June 2006.
3. Ronan Cummins and Colm O'Riordan. An axiomatic study of learned term-weighting schemes. In *SIGIR'07 workshop on learning to rank for information retrieval (LR4IR-2007)*, pages 11–18, July 2007.
4. Weiguo Fan, Michael D. Gordon, and Praveen Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing & Management*, 2004.
5. Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487. ACM Press, 2005.
6. H. S. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Inc., Orlando, FL, USA, 1978.
7. N. Oren. Re-examining tf.idf based information retrieval with genetic programming. *Proceedings of SAICSIT*, 2002.
8. Nir Oren. Improving the effectiveness of informtation retrival with genetic programming. Master's thesis, Faculty of Science, University of the Witwatersrand, South Africa, December 2002.
9. Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *In D. K. Harman, editor, The Third Text REtrieval Conference (TREC-3) NIST*, 1995.
10. Andrew Trotman. Learning to rank. *Information Retrieval*, 8:359 – 381, 2005.

Evolutionary simulations of behaviours in a common-pool resource problem

Dara Curran[1], Colm O'Riordan[2] and Humphrey Sorensen[1]

[1] Department of Computer Science, University College Cork
[2] Department of Information Technology, National University of Ireland, Galway
d.curran@cs.ucc.ie,
colm.oriordan@nuigalway.ie,
h.sorensen@cs.ucc.ie

Abstract. Issues regarding foraging in groups have been addressed and researched in a range of domains. Questions arise regarding the benefits to the group as a whole and the cost placed upon individual group members. In this paper, we model the foraging problem as a common resource pool problem and evolve populations in a range of scenarios. In these simulations, agents (group members) forage for food, may contribute to a common pool resource and may benefit from this group resource. We present and discuss results illustrating the scenarios under which agents evolve to behave for the common good of the group and its effect on the survival and the fitness of the population.

1 Introduction

The importance of group behaviours has been addressed in a range of domains including, among others, biology, anthropology, ecology and artificial life. Most species exhibit some form of group behaviour. The advantages and benefits of group membership include, among others, increased anti-predator vigilance[14], conservation of energy[1], dilution of risk[15] and foraging benefits[4][13][17]. There are also potential costs associated with group membership; examples include potential theft of individual resources (klepoparasitism)[2] and interference in foraging[12].

Several of the oft-cited examples of group behaviour represent social dilemma problems where the optimal behaviour for the group differs from behaviours that are best for the individual. Free-rider problems abound whereby individuals may reap the benefit of group membership while avoiding the costs. The task of foraging and hunting represents one such free-rider problem. In many cases, foraging as a group confers an advantage to the members in the group. However, there is a temptation to reap the rewards while attempting to avoid the costs.

There are many examples of group behaviour whereby members choose not to free ride and instead choose the behaviour that is collectively rational [17].

Several of these examples can be modelled as a common resource pool problem, where participants may contribute to a common pool and may also utilise this pool when required.

In this paper, we evolve populations of agents participating in foraging tasks (in different environments). In this simulator, agents may contribute to a given pool which may be used by all members.

The paper is structured as follows. The following section briefly discusses some related work. Section 3 discusses our model and experimental setup. The subsequent sections presents results, discussions and conclusions.

2 Related Work

2.1 Foraging and Groups

The majority of work undertaken in the study of foraging in groups has involved empirical studies of species and their behaviours. There have been several studies on particular examples of cooperative group behaviour adopted in foraging. These have involved: individuals working together to attack and kill a prey that individually would have been impossible to kill [5]; sharing information regarding the location of prey or food[16]; and sharing obtained food with other members[17]. An oft-cited example of food sharing by contributing to a common pool is that of the Aché tribe[7] whereby hunters donate all their gains to a common pool and exclude themselves from sharing in their own hunted prey.

2.2 Social Dilemmas and Common Pool resource problems

Problems inherent in group foraging can be viewed as social dilemmas and free-rider problems. In these problems, all group members benefit from whatever utility or gain is earned by the group. However those that do not contribute to the common pool effectively gain the most by not expending energy in contributing. A well-known example is the *Tragedy of the Commons*[8]. In this dilemma, land (the commons) is freely available for farmers to use for grazing cattle. For any individual farmer, it is advantageous to use this resource rather than their own land. However, if all farmers adopt the same reasoning, the commons will be over-used and soon will be of no use to any of the participants, resulting in an outcome that is sub-optimal for all farmers.

This is has been modelled in many ways including the N-player prisoner's dilemma and as common pool resource problems. In previous work, researchers have investigated means which induce the optimal outcome for the group. Previous approaches include spatial constraints[9][3][6], tagging mechanisms[11] and trust and reputation systems[10].

3 Model

The model consists of a population of agents that inhabit an environment where they must forage for food to survive. Each agent is born with a certain amount of energy that is expended during its life. The agent has a number of foraging opportunities, each of which cost the agent one unit of life energy. If the agent

successfully finds food, its life energy is augmented. The likelihood of an agent successfully finding food is controlled by a simple probability. This is a global probability, meaning that all agents have the same probability of finding food.

Each agent also has an opportunity to donate foraged food to a common pool. The amount of food that an agent will donate to the pool is determined by its genetic makeup. Each agent has a chromosome of 20 bits which corresponds to the amount of food that that agent will donate to the pool at the end of all foraging turns.

Once all agents have foraged for the given number of foraging turns, the pool is redistributed across the population. However, the pool is also allowed to grow during this time, meaning that the amount distributed at the end of each generation is larger than the amount that was placed in the pool. An agent's fitness is directly linked to its life energy and therefore, to the amount of food it has been able to forage, plus the slice of the pool it receives. Mating opportunities are proportional to fitness, but if an agent's life force reaches 0 the agent is considered to have died and is not capable of reproducing. It is therefore possible for an entire population to become extinct.

The division of the common pool can be implemented in a number of ways. In these experiments, we analyse the effects of two methods. The first, the equal distribution method, divides the pool equally between those agents that are still alive at the end of a generation. The second, the proportional distribution method, divides the pool proportionally - in other words, an agent receives a portion of the pool proportional to the amount of food it contributed.

The two distribution types equate to a population sharing its resources among the whole population (benefiting free-riders) in the case of equal distribution, and a population where resources are only given to those that contributed to the resource pool. Thus the model can be seen as examining the effects of sharing on populations.

In addition, we wish to examine the effect of altering the foraging success of the population. This is also implemented in two ways. In the first we run a number of separate experiments varying the foraging success probability from 0.2 to 0.8. In the second, we use a single population and we shift the foraging success from 0.2 to 0.8 and back during the course of the experiment. Thus, the first represents a series of static environments and the second a dynamic environment.

Each set of results is averaged from 20 independent runs. The static environment experiments are allowed to run for 500 generations, while the dynamic environment experiment run for 2500 generations. Each use a population of 250 agents. Each generation of agents is allowed 10 foraging turns and the reward for a successful foraging is 3 units of life energy. Finally, the common pool grows to three times its size at each generation.

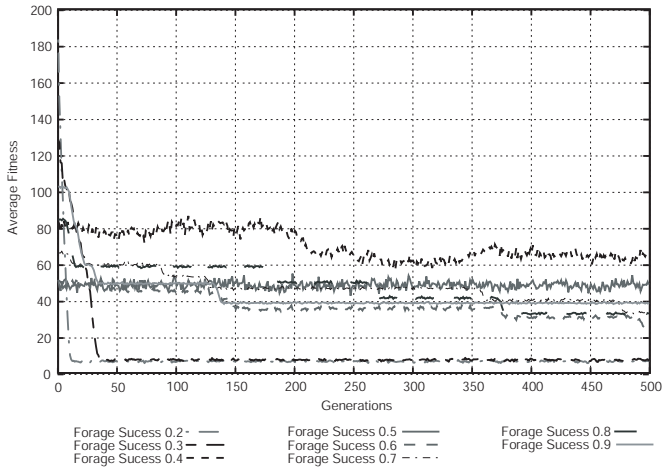


Fig. 1. *Average fitness in equal distribution population*

4 Static Environments

The first set of experiments examine the effects of different forage success probability values on two populations: one distributing the common pool equally and the other distributing it proportionally.

4.1 Equal Distribution

The first set of results illustrated in Figure 1 shows the fitness of each population over time. As one would expect, the populations with the lowest probability of foraging success have the lowest fitness. However, the populations with the highest probability of success do not have the highest fitness. Instead, it is population with a success probability of 0.4.

The second figure illustrates the size of each population over time (Figure 2). The largest populations are those with the highest foraging probability, although even populations with lowest success probabilities are capable of maintaining up to 80% of the original population. Therefore, the situation for the populations with higher success probabilities is that they have large populations, but only mediocre fitness. In contrast, the population with the highest fitness (success probability 0.4) has a relatively small population.

The third set of results is the average percentage donators in the population, illustrated in Figure 3. The populations with the fewest donators are those with the lowest probabilities of foraging success (0.2 to 0.3). Again, this is understandable as they are not capable of finding food often, they are unlikely to donate food to the common pool.

Populations with high foraging success (0.7 to 0.9) begin with many donators but then decline over the course of the experiment. Clearly, the benefit received

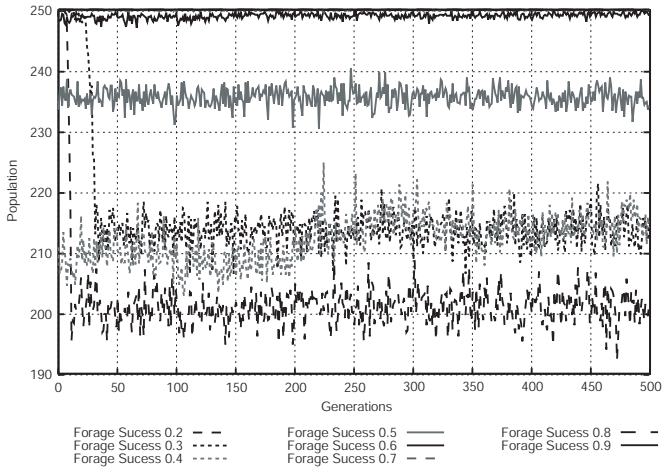


Fig. 2. *Average population size in equal distribution population*

from the common pool does not justify the expense of donating food and these populations begin to lose donators.

The populations with the highest number of donators are those that are exposed to foraging success approaching the median (0.4-0.5) and infact, the highest of all is the population with foraging success of 0.5. Clearly, for these populations it is advantageous to donate to the pool because the likelihood of unsuccessful foraging is precarious enough for the common pool to be useful. Equally, the likelihood of consistently finding food is low enough to allow the common pool to be useful to successful foragers.

The final set of results shows the average amount donated by each population and are illustrated in Figure 4. Once again, the same patterns emerge, with the populations with the lowest probabilities of success donating the least, followed by the populations with the highest probabilities of success. The most generous population in terms of donations is the population with foraging success of 0.4.

To summarize, populations with high probabilities of foraging success tend to be large, but with only mediocre fitness. While they initially contain a high number of donators, this decreases dramatically over time and in any case, the donations given to the common pool tend to be small. Populations with low probabilities of foraging success tend to be relatively small, with low fitness and contain few donators which donate very little to the common pool.

The most interesting results are those from populations with foraging success probabilities that are around 0.4-0.5. These have smaller, more fit populations that contain relatively many donators which donate generously to the common pool. Thus, while these populations may not be able to sustain as many individuals as those with more foraging success, it would seem that the individuals are fitter on average (clearly as a result of the redistribution of the common pool).

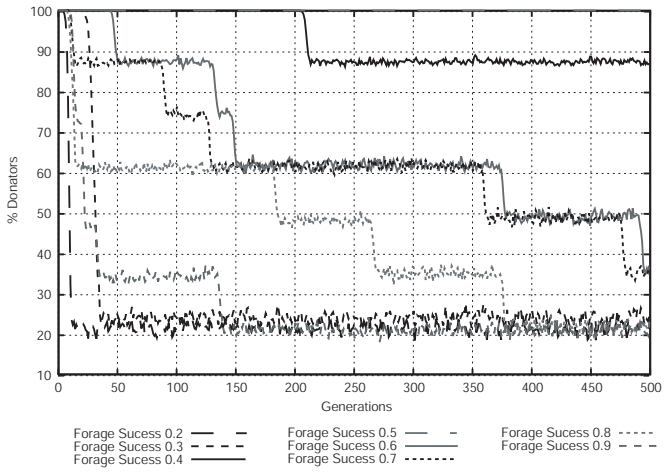


Fig. 3. *Average percentage donators in equal distribution population*

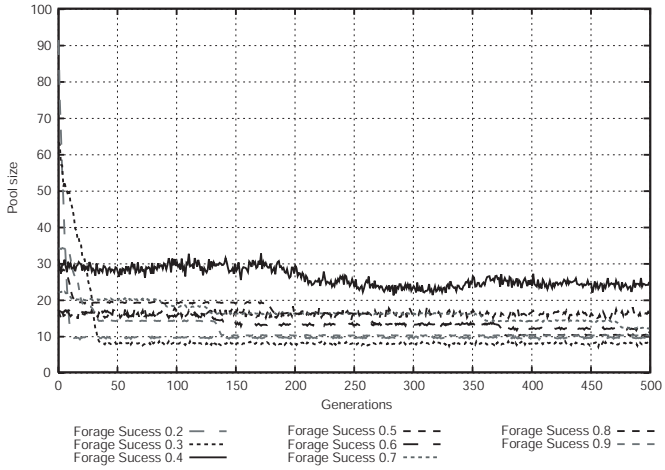


Fig. 4. *Average amount donated in equal distribution population*

4.2 Proportional distribution

The following set of results examine the effect of distributing the common resource pool among individuals in proportion to the donation an individual has made. Thus, individuals that do not contribute to the pool do not receive a share of the pool.

The first set of results, illustrated in Figure 6, shows the fitness of each population over time. In contrast to the experiments where the common pool is distributed equally across the population, the fitness of each population is highly dependent on the forage success of each population. The most successful population from the point of view of fitness is the one with a forage success of 0.9. This is followed by the populations with 0.8, 0.7 and so on. Thus, it can be said that the common pool does not seem to have a large effect on the fitness of the population, as the populations stratify according to forage success, just as one would expect if no distribution were employed.

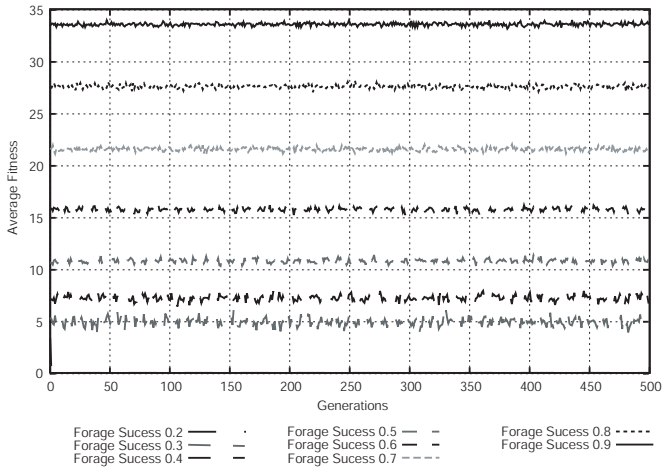


Fig. 5. *Average fitness in proportional distribution population*

The second set of results, examining population size, are illustrated in Figure 6. The largest populations are those with highest forage success (0.6-0.9). Even the population with only 0.5 forage success is capable of sustaining a relatively large population. The worst performing population is that with forage success of 0.2, which becomes extinct almost immediately. There is also a considerable difference between the population with forage success of 0.3 and that with 0.4.

To understand these results, they must be examined in conjunction with the amount of resources each population is contributing to the common pool. This is illustrated in Figure 7. The first interesting aspect of these results is that the population with forage success of 0.3 donates the largest amount of resources to the common pool by a large margin. In contrast, the population with forage

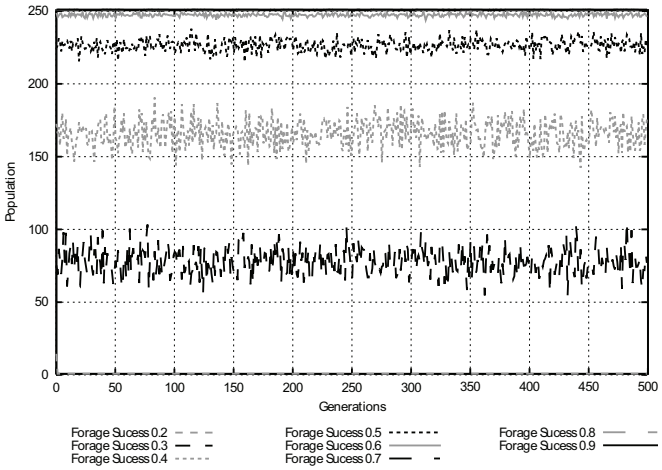


Fig. 6. *Average population size in proportional distribution population*

success of 0.2 is tied with the population with forage success of 0.6 with the lowest contributions.

The difference between the 0.2 and 0.3 populations is striking. Clearly, the risk of a population becoming extinct is high with a forage success of 0.2 and is only slightly better with a forage success of 0.3. The population with forage success of 0.3 is capable of foraging slightly more food on average than that of the population with 0.2 forage success. However, if population invests heavily in the common resource pool, it will be able to counter some of the effects of unsuccessful foraging.

The remaining populations are clustered quite closely at a moderate level of donation. Thus the most successful populations are those with high foraging success. These tend to donate only moderately to the common resource pool. The most interesting population is that with forage success of 0.3, which is capable of staving off extinction only through heavy investment into the common resource pool. The graph for the percentage donators in each population is not shown because in each case, the percentage was 100%. In other words, in each population every individual contributes something to the pool.

5 Discussion/Conclusion

The two experiments examine the difference between populations where a common resource pool is distributed equally, and one where the pool is distributed in proportion to the amount each individual donated. There are a number of striking differences between the two populations. Firstly, the only extinction occurs in the population employing proportional distribution. Not only do no extinctions occur in the equal distribution populations, but the size of the smallest population is around 200 individuals, only 20% smaller than the maximum.

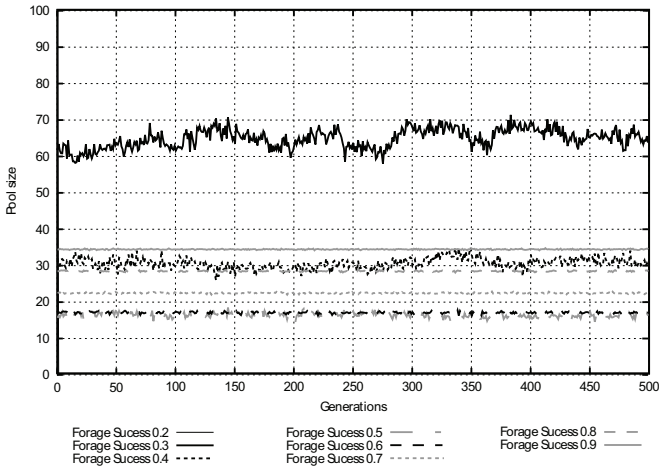


Fig. 7. *Average amount donated in proportional distribution population*

In populations employing equal distribution, most populations converge towards small donations given by a relatively small number of individuals. By contrast, the populations employing proportional distribution, every individual tends to donate larger amounts.

Finally, the fitness of the two populations is very different. The average fitness of each population is considerably smaller in populations using proportional distribution than in those that share the pool equally. Furthermore, the fitness levels associated with the population employing proportional distribution are stratified according to forage success. Thus it is likely that the population as a whole gains little, if anything, from the distribution of the common resource pool.

As a model of group foraging, our model explores outcomes in a range of scenarios. Real world examples are far more complex than the abstract view adopted in our work and the types of foraging are influenced by a number of factors not considered in our model e.g. status of the individual in the society, the presence of relations between members. Future work will involve further investigation of foraging societies. In particular, we wish to extend the model further to explain reported phenomena in the existing empirical studies.

References

1. R.V Andrews and R.W Belknap. Bioenergetic benefits of huddling in deer mice (*peromyscus maniculatus*). *Comparative Biochemistry and Physiology A*, 85:775–778, 1986.
2. M. Broom and G. D. Ruxton. Evolutionarily stable kleptoparasitism: consequences of different prey types. *Behavioural Ecology*, 14(1):23–33, 2003.
3. Hauert C. Spatial effects in social dilemmas. *Journal of Theoretical Biology*, 240(4):627–36, June 2006.

4. S. Creel. Cooperative hunting and group size: assumptions and currencies. *Animal Behaviour*, (54):1319–1324, 1997.
5. S. Creel and N.M Creel. Communal hunting and pack size in african wild dogs. *Animal Behaviour*, (50):1325–1339, 1995.
6. U. Dieckmann, R. Law, and J.A.J. Metz, editors. *Games on grids*, pages 135–150. Cambridge University Press, 2000.
7. M. Gurven. Reciprocal altruism and food sharing decisions among Hiwi and Ache huntergatherers. *Behavioral Ecology and Sociobiology*, 56(4):366–380, 2004.
8. Garret Hardin. The tragedy of the commons. *Science*, 162:1243–1248, December 1968.
9. Kristian Lindgren and Mats G. Nordahl. Evolutionary dynamics of spatial games. *Phisica D*, 75:292–309, 1994.
10. S. Ramchurn, D. Huynh, and N. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.
11. R. L. Riolo. The effects of tag-mediated selection of partners in evolving populations playing the iterated prisoner's dilemma. Technical report, Santa Fe Institute Working Paper 97-02-016, 1997.
12. G.D Ruxton. Foraging in flock-nonspatial models may neglect important costs. *Ecological Modelling*, 82:235–53, 1993.
13. R.J Schmitt and S.W Strand. Cooperative foraging by yellowtail (*seriola lalandei*) on two species of fish. *Copeia*, pages 714–717, 1982.
14. J.E. Treherne and W.A Foster. Group transmission of predator avoidance behaviour in a marine insect. *Animal Behaviour*, (28):1119–1122, 1980.
15. G.F Turner and T.J Pitcher. Attack abatement: a model of group protection by combined avoidance and dilution. *American naturalist*, 35:228–240, 1986.
16. T.J Valone. Group foraging, public information and patch estimation. *Oikos*, (56):357–363.
17. Gerald S Wilkinson. Reciprocal food sharing in the vampire bat. *Nature*, (308):181–184, 1984.

Evolving Team Behaviours in Environments of Varying Difficulty

Darren Doherty and Colm O'Riordan

Department Of Information Technology
National University Of Ireland, Galway.
darren.doherty@nuigalway.ie
colm.oriordan@nuigalway.ie

Abstract. This paper investigates how varying the difficulty of the environment in a 2-D combative gaming setting can affect the evolution of team behaviour. The difficulty of the environment is altered by varying the field of view and viewing distance of the agents. The behaviours of the agents are evolved using genetic programming (GP) techniques. These experiments show that the level of difficulty of the environment does have an impact on the evolvability of effective team behaviours; i.e. simpler environments are more conducive to the evolution of effective team behaviours than more difficult environments.

1 Introduction

In this paper, we investigate how the difficulty of the environment affects the evolution of effective team behaviours in a combative 2-D gaming setting. The environmental difficulty is varied by altering the field of view and viewing distance of the agents; thereby, varying their ability to perceive information from the environment. Throughout these experiments, the same evolutionary algorithm (EA) is used to examine the effect of varying the difficulty of the environment on the fitness of the evolved teams. In previous work, we have evolved successful teams in an environment where agents had perfect vision through a field of 180 degrees. In this work, we wish to test the robustness of this approach by varying the environmental difficulty. For these experiments, we explore team agents' ability to evolve in eight different environmental settings. These range from very restrictive environments, with short viewing distances and narrow fields of view, to very unrestricted environments, with long viewing distances and wide fields of view.

We hypothesise that the shorter the viewing distance and the narrower the field of view, the more difficult it will be for the agents to evolve successful behaviours as their perceptual abilities are more limited. Additionally, larger viewing distances and wider angles enable the agents to perceive more of their environment, which should be more conducive to the evolution of effective behaviours. We believe that these latter experiments should produce results comparable to our previous experiments in which agents had perfect vision through

a field of 180 degrees. Intermediate viewing distances may produce mediocre results as agents may find difficulty locating the enemy, weapons and health packs on their own.

Another motivation for choosing to vary the perceptual capabilities of agents to alter the level of difficulty of the environment is that of realism, as humans have a limited field of view and viewing distance within which they can successfully classify objects. Altogether, eight experiments are undertaken over four viewing distances and two fields of view and the results analysed.

In the remainder of this paper, the simulation environment, game agents and the genetic program to be used in the evolution will be discussed, together with the experimental setup and a discussion of the results of the experiments.

2 Related Work

In the past, GP techniques have been successfully used to evolve tactics and strategies for groups of computer controlled agents in a number of different domains. Richards et al. used a genetic program to evolve groups of unmanned air vehicles (UAVs) to effectively and efficiently search an uncertain and/or hostile environment [10]. Three different levels of environmental difficulty were used in their experiments. The first two environments had rectangular search areas but the second also had a single fixed hostile agent making it more difficult. The third environment has two irregular shaped search areas and a no fly zone making it the most difficult environment of the three. Their results show a correlation between the fitness of the evolved flying strategies and the difficulty of the environment. Pursuit strategies for predator-prey domains have also been evolved using GP techniques [6, 8]. Luke et al. evolved predator strategies that enable a group of lions to successfully hunt gazelle [8]. In order to vary the difficulty of the environment, the lions were given different levels of sensing ability in each of the experiments. This is a similar method of varying environmental difficulty to the one used in this paper. In addition, GP has been used to successfully evolve sporting strategies for teams of volleyball players [9] and teams of soccer players [7]. Moreover, tactics for teams of armed forces in a combative, hostile setting have been successfully evolved using GP techniques [3–5].

3 Simulation Environment

The environment used in this work is similar to that used in previous research [4, 5] (see Fig. 1). It consists of an open 2-dimensional space, enclosed by four walls and is built on the 2-D *Raven* game engine [1]. The evolved team (consisting of five agents) is pitted against a single powerful enemy agent. Items are placed on the map at locations that are equidistant from both the team starting points and the enemy starting point. These items consist of health packs and a range of weapons.

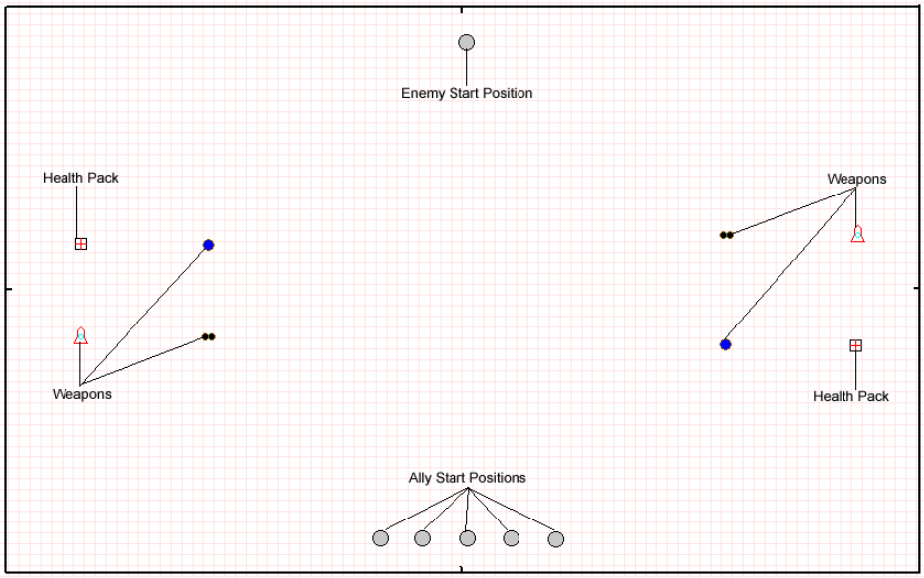


Fig. 1. Simulation Environment Map

4 Game Agents

There are two types of agents in the simulation environment: team agents and the single enemy agent. Both types of agent use the same underlying goal-driven architecture to define their behaviour [2]. However, the methods used to decide which goal to pursue at any given time are different. The single enemy agent uses hand-coded desirability algorithms associated with each goal to decide on its course of action, whereas team agents use an evolved decision-making tree to decide which goal to pursue at any given time.

Team agents begin the game with the least powerful weapon in the environment but have unlimited ammunition for it, so agents always have the ability to attack. The single enemy has five times the health of a team agent and the most powerful weapon in the environment with infinite ammunition.

All agents have a limited range within which they can visually perceive information from their environment. An agent's visual capability is defined by a field of view and a viewing distance. The fields of view of both agent types are equal but the viewing distance of the enemy agent is twice that of the team agents.

Both types of agent have a memory which allows them to remember information they perceive from their environment. Any dynamic information, such as ally or enemy positions, is forgotten after a specified time (five seconds for these experiments). However, static information, such as the location of items, is not forgotten if perceived, as their location does not change for the duration of the game.

As there is only one enemy agent, team agents do not have to worry about selecting a target they wish to attack. The enemy agent chooses its target based on distance, so if more than one team agent has been recently sensed, the agent closest to the enemy is targeted.

5 The Genetic Program

The genetic program used to evolve the decision-making trees for the team agents is similar to that used in our previous research [4, 5]. The entire team of five agents is viewed as one chromosome, so team fitness, crossover and mutation operators are applied to the team as a whole. Each of the five agents are derived from a different part of the team chromosome, so evolved teams are heterogenous.

A strongly typed genetic program (STGP) is adopted. The same five node sets applied in previous research [4, 5] are used: action, conditional, numerical, environmental and positional. However, a number of new nodes have been added to the node sets in order to accommodate the limited viewing capabilities of the agents. For example, *can_see_enemy* node is added to allow an agent to make decisions based on when the enemy has become visible to the agent. In total, there are fifty nodes across the five node sets that can be used to comprise a GP tree.

The fitness function takes into account: the length of time a game lasts, the remaining health of both the enemy agent and ally team and the length of the chromosome (to prevent bloating of the trees).

$$RawFitness = \frac{AvgGameTime}{Scaling * MaxGameTime} + \frac{(5 * (Games * TSize * MaxHealth - EH) + AH)}{Games * TSize * MaxHealth}$$

where $AvgGameTime$ is the average duration of the evaluation games, $Scaling$ is a variable to reduce the impact of game time on fitness, EH and AH are the total amount of health remaining for the enemy agent and for all five team agents respectively, $TSize$ is the number of agents in the evolving team (i.e. five), $Games$ is the number of games played per evaluation (in this work the number of games per evaluation is set to twenty) and $MaxHealth$ is the maximum health an agent in the game can have.

More importance is attached to the change in the enemy agent's health than the corresponding change in the team's health as the tactics are evolved to be capable of defeating the enemy. Longer games are also favoured to prevent teams who are capable of surviving the enemy's attack from dying off in the earlier generations.

The length of the chromosome is taken into account and fitness values are then inverted using the following formula such that values closer to zero are better.

$$StdFitness = (MaxRF - RawFitness) + \frac{Length}{LengthFactor}$$

where $MaxRF$ is the maximum value possible $RawFitness$ can hold and $LengthFactor$ is a parameter used to limit the influence the length of the chromosome has on the fitness.

Team selection, crossover and mutation operators are identical to those used in our previous work. For a detailed explanation of selection, crossover and mutation operators used see [4, 5].

6 Experimental Setup

In these experiments, we investigate how varying the difficulty of the environment in a 2-D combative gaming setting can affect the evolution of team behaviour. The difficulty of the environment is varied by altering the evolving agents' perceptual capabilities. Experiments are set up for two fields of view (90 and 180 degrees) and four viewing distances (50, 200, 350 and 500 pixels). Eight experiments are used to investigate the evolution of team behaviours in these environments. Maximum and minimum viewing distances can be calculated based on the size of the grid. The maximum viewing distance of the agents in the experiments is set to 500 pixels and the minimum viewing distance is set to 50 pixels for these experiments. Intermediate distances are then chosen as equally spaced distances between 50 and 500 pixels.

The enemy viewing distance is scaled relative to the viewing distance of the team agents. As there are five team agents and only the one enemy agent, the collective viewing range of the team covers a much larger portion of the map than that of a single agent. Therefore, it was decided to allow the enemy's viewing distance to be twice that of a team agent. Note that for a viewing distance of 1000 pixels, the enemy will have perfect vision within its field of view as 1000 pixels is greater than both dimensions of the map.

Twenty separate evolutionary runs are performed in each of the eight environments. In each of the runs, 100 team chromosomes are evolved over 100 generations. Each team evaluation comprises twenty games. The best performing team from each of the runs is recorded.

Each recorded team is then tested more extensively using a larger number of games to obtain a more accurate measure of its performance. The validation tests involve evaluating each recorded team's performance over 1000 games and basing the team's fitness score on their average performance over the 1000 games. The number of games won, lost and drawn are also recorded. Once these tests are performed for each of the recorded teams, the maximum, minimum, average and standard deviation of the team's fitness and of the number of wins, losses and draws are found for the twenty recorded teams in each of the eight experiments.

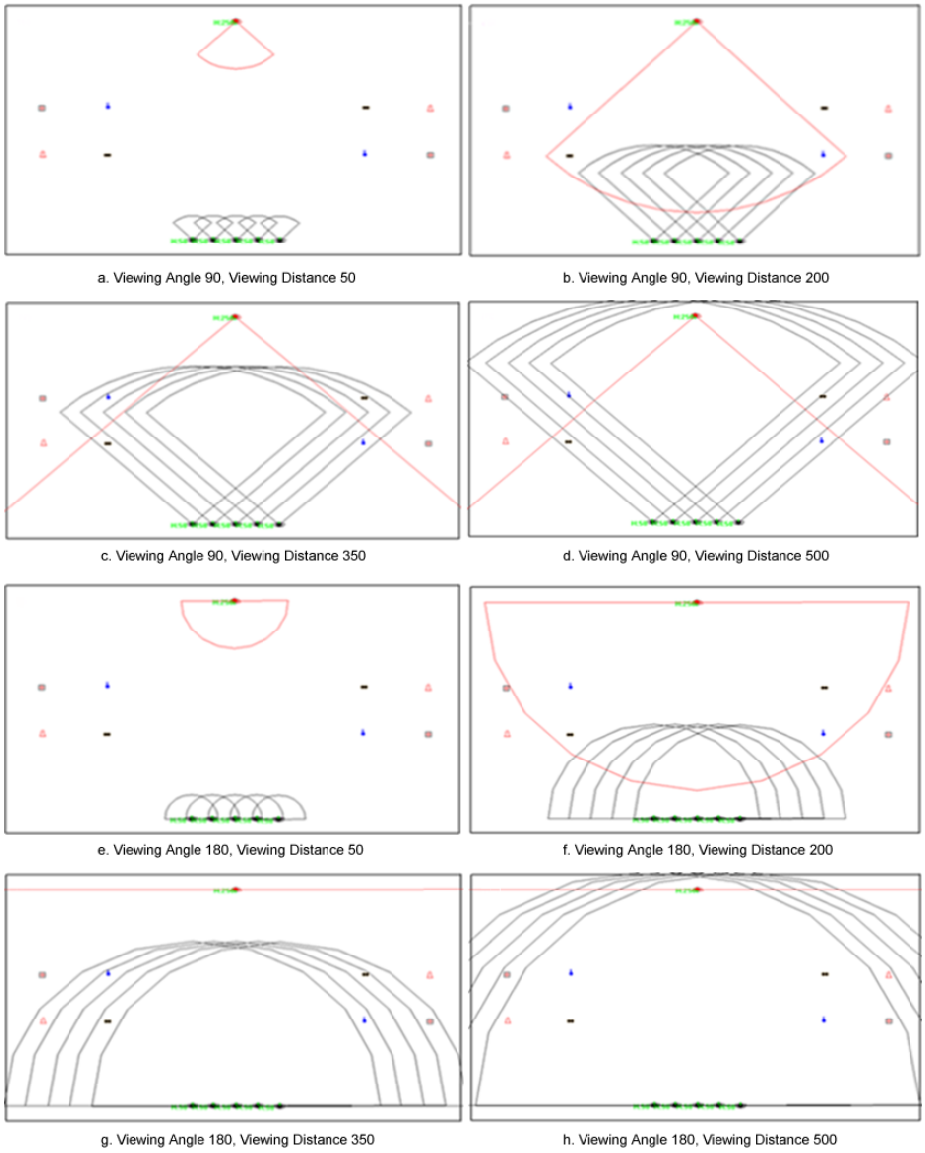
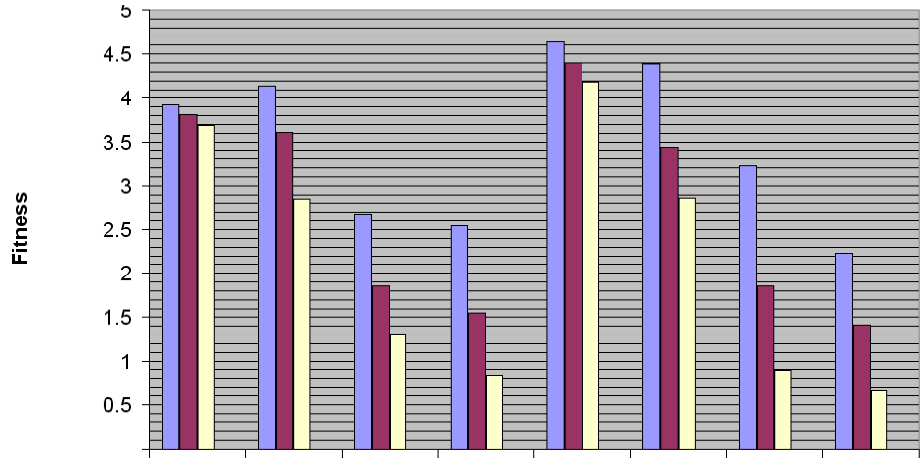


Fig. 2. Experimental setups for various fields of view and viewing distances

These results are then analysed to see if environmental difficulty has a significant impact on the evolution of effective team behaviour.

7 Results

The results of the experiments show that the level of difficulty of the environment does have an impact on the EA's ability to evolve effective team behaviours. Fig. 3 shows the maximum, minimum, average and standard deviation of fitness for the twenty runs in each of the eight environments over 1000 games.



Field of View	90	90	90	90	180	180	180	180
Viewing Distance	50	200	350	500	50	200	350	500
Worst	3.92	4.14	2.68	2.54	4.64	4.39	3.22	2.23
Average	3.81	3.60	1.85	1.55	4.40	3.44	1.85	1.42
Best	3.68	2.85	1.31	0.84	4.18	2.86	0.89	0.67
Std Dev	0.06	0.34	0.37	0.55	0.09	0.33	0.65	0.45

Fig. 3. Validation results for team fitness over 1000 games

Note that fitness values closer to zero are better. As is evident from Fig. 3, there is a correlation between the agent's viewing range and the GP's ability to evolve effective team behaviours.

Fig. 4 shows the number of games won by the best performing recorded team in each of the eight environments. The results follow a similar trend for both, the environments where the field of view is 90 degrees and environments where the field of view is 180 degrees. This is understandable as the field of view of both the enemy and team agents are equivalent in each of the eight experiments. Thus, varying the field of view does not have as much of a bearing on the difficulty of the environment as varying the viewing distance. Generally, as the level of difficulty of the environment decreases, the number of games won by the best evolved team increases. This follows a similar trend to the fitness values shown in Fig. 3. The best fitness in each of the environments gradually improves as the environment becomes less difficult, i.e. as the viewing distance of agents increase.

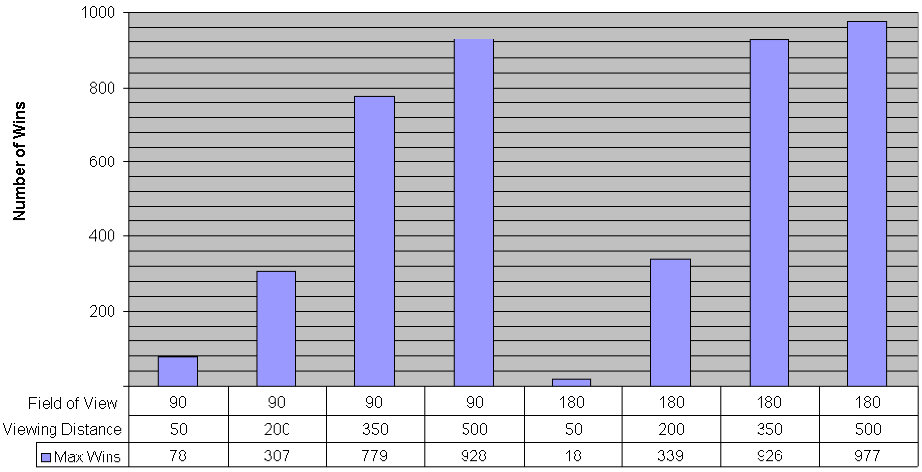
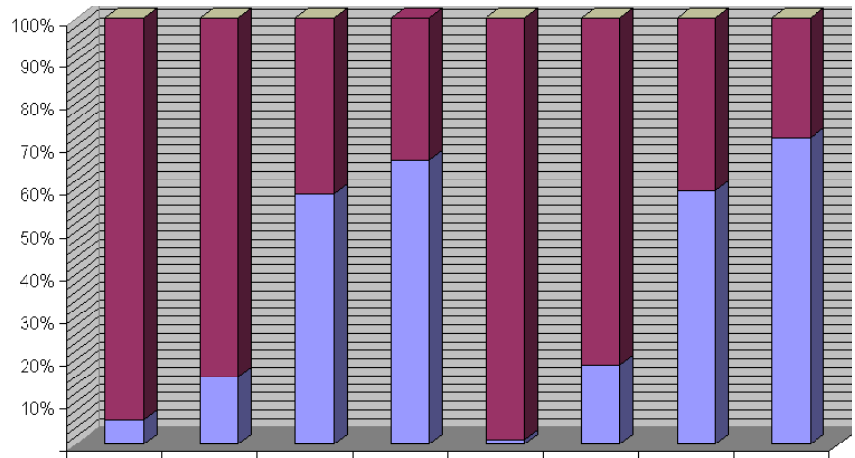


Fig. 4. Maximum number of games won by a team in each environment

In the most difficult environments, where the viewing distance of agents is 50 pixels, the maximum number of games won by any of the recorded teams is less than 8% of games played, less than 80 out of 1000 games. In contrast, the maximum number of games won by the best evolved team in the least difficult environments, is over 97% of games played.

As mentioned earlier, the results are comparable for both environments with 90 and 180 degree fields of view, as there is a steady increase in the number of games won as the viewing distance of the agents in the environment increases. Overall, the results of the experiments with the 180 degree field of view are generally slightly better, with the exception of the experiment where the viewing distance is 50 pixels. We believe this is because 50 pixels is too restrictive of a distance to benefit from the larger viewing angle but the enemy's viewing distance of 100 pixels in this environment can benefit greatly from the larger field of view making it easier for the enemy to dispose of the team. Hence, this environment can be regarded as the most difficult of all eight environments.

Fig. 5 shows the average number of wins, losses and draws, out of 1000 games for all twenty recorded teams in each of the eight environments. We can clearly see a steady trend in the average number of wins, losses and draws of teams as the environmental difficulty changes. In both, environments where team agents have a field of view of 90 degrees and environments where the agents' field of view is 180 degrees, there is a steady increase in the average number of wins of the evolved teams as the level of difficulty of the environment decreases.



Field of View	90	90	90	90	180	180	180	180
Viewing Distance	50	200	350	500	50	200	350	500
Draws	1.05	0.45	0.05	0.00	0.25	0.35	0.15	0.05
Losses	942.35	843.05	414.10	333.50	988.25	815.15	405.90	283.90
Wins	56.60	156.50	585.85	666.50	11.50	184.50	593.95	716.05

Fig. 5. Average number of games won, lost and drawn by all recorded teams

8 Conclusions

In this paper, we explored how varying the level of difficulty of the environment can affect the evolution of effective team behaviours in a combative 2-D gaming environment. We discussed the simulation environment, game agents and genetic program to be used in the evolution, together with the experimental setup and the results of the experiments.

The results displayed in Fig. 3, Fig. 4 and Fig. 5, clearly indicate that the level of difficulty of the environment affects the evolution of effective team behaviours. The performance of evolved teams improves as the difficulty of the environment decreases. In the very difficult environments, where agents' viewing range is very restricted, teams perform very poorly. The EA is unable to evolve good solutions as the environment is too difficult. In contrast, in the less difficult environments, the evolved teams perform very well as the EA is able to evolve effective team behaviours.

In future experiments, we wish to explore the effects communication between agents could play in the evolution of team behaviours, as communication would be potentially very useful in environments where agents' viewing range is more restricted. Agents could share information and perceive the environment as a team rather than individually. The position of the enemy or other game objects could be communicated amongst team agents, which could allow for much more effective team behaviours to evolve. We also wish to analyse the actual

behaviours evolved to see if there are common properties in the behaviours of teams evolved in similar environments.

Acknowledgment

The primary author would like to acknowledge the Irish Research Council for Science, Engineering and Technology(IRCSET) for their assistance through the Embark initiative.

References

1. Mat Buckland. *Programming Game AI by Example*, chapter Raven: An Overview, pages 295–333. Wordware Publishing, Inc, 2005.
2. Mat Buckland. *Programming Game AI by Example*, chapter Goal-Driven Agent Behaviour, pages 379–415. Wordware Publishing, Inc, 2005.
3. Darren Doherty and Colm O'Riordan. Evolving agent-based team tactics for combative computer games. In *AICS 2006 17th Irish Artificial Intelligence and Cognitive Science Conference*, 2006.
4. Darren Doherty and Colm O'Riordan. Evolving tactical behaviours for teams of agents in single player action games. In *CGAMES 2006 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, 2006.
5. Darren Doherty and Colm O'Riordan. A phenotypic analysis of gp-evolved team behaviours. In *GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1951–1958, New York, NY, USA, 2007. ACM Press.
6. Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Sandip Sen, editor, *International Joint Conference on Artificial Intelligence-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada, 20-25 1995. Morgan Kaufmann.
7. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *International Joint Conference on Artificial Intelligence-97 First International Workshop on RoboCup*, Nagoya, Japan, 1997.
8. Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 1996. MIT Press.
9. Simon Raik and Bohdan Durnota. The evolution of sporting strategies. In Russel J. Stonier and Xing Huo Yu, editors, *Complex Systems: Mechanisms of Adaption*, pages 85–92, Amsterdam, Netherlands, 1994. IOS Press.
10. Marc D. Richards, Darrell Whitley, J. Ross Beveridge, Todd Mytkowicz, Duong Nguyen, and David Rome. Evolving cooperative strategies for uav teams. In *GECCO '05: Proceedings of the 7th Annual conference on Genetic and evolutionary computation*, pages 1721–1728, New York, NY, USA, 2005. ACM Press.

Using User Model Information to Support Collaborative Filtering Recommendations

Josephine Griffith[1], Colm O'Riordan[1], and Humphrey Sorensen[2]

[1] Dept. of Information Technology,
National University of Ireland, Galway, Ireland
josephine.griffith@nuigalway.ie, colm.oriordan@nuigalway.ie
[2] Dept. of Computer Science, University College Cork, Cork, Ireland
sorensen@cs.ucc.ie

Abstract. This paper considers some of the information that can be captured about users and groups from a collaborative filtering dataset with the aim of using this information to provide a more personalised recommendation experience. The idea is that features of users are used to identify when a set of recommendations are formed using particularly weak evidence or particularly strong evidence. This evidence will be returned to a user together with a set of recommendations and will give the user more information with which to judge if the recommendations are likely to be accurate or of interest to the user, e.g., on occasion the user may choose to discard the recommendations if they have been produced by the system using a "weak" set of evidence.

1 Introduction

Collaborative filtering systems automate the "word of mouth" process that commonly occurs within social networks [15], i.e. people will seek recommendations from people with whom they share similar preferences in an area. Within the field of collaborative filtering many models and techniques have been proposed, tested and compared. Additional features of users, items and the recommendation task have also been considered (e.g., product information, demographic information, time, trust). Studies involving new models, techniques and incorporating additional information often have different foci where the aim has not always been to improve performance. For example, various studies have focused on dealing with scalability issues [5], dealing with the issue of dataset sparseness [7], including additional information [1], incorporating trust [11] and dealing with noise [12].

Although collaborative filtering is most frequently seen as a way to provide recommendations to a set of users, collaborative filtering datasets also allow for the analysis of social groups and of individual users within a group, thus providing a means for creating a new user model, group model or for augmenting an existing user or group model. We believe that such analysis can also be used to provide evidence of how accurate the system predictions are. Mirza et al. [10] list four desirable aspects of recommendation:

1. “Recommendation is an indirect way of bringing people together.”
2. “Recommendation, as a process, should emphasize modeling connections from people to artifacts, besides predicting ratings for artifacts.”
3. “Recommendations should be explainable and believable.”
4. “Recommendations are not delivered in isolation, but in the context of an implicit/explicit social network.”

The approach taken here concentrates on making recommendations more “explainable and believable” by providing users with information relating to whether a recommendation may be accurate or not. The motivation for this work is that although users are often clustered into groups based on nding “similar users” and much is known about the e ect of various user, item and group features on the accuracy of predictions, this information has not been used to support the output of recommendation systems. Although it is unlikely that sufficiently clear evidence can be found to support all user recommendations, it could still be useful to highlight the cases when a set of recommendations are formed using particularly weak evidence or particularly strong evidence.

In this work, six features that can be extracted from the collaborative ltering dataset are rstly identi ed, de ned and analysed with respect to their e ect on recommendation accuracy. Some of these features are particular to the recommendation task while some features use measures from social network theory and information retrieval. Each feature can provide one piece of “evidence” if its values are above or below a certain threshold. Thresholds are chosen based on the analysis of the e ect of the features on prediction accuracy. The more positive or negative pieces of evidence that exist for a given user, the more likely that the recommendation results will be accurate (for positive evidence) or inaccurate (for negative evidence). The paper outline is as follows: Section 2 presents related work in collaborative ltering. Section 3 outlines the methodology, presenting the collaborative ltering approach, specifying the user features which are extracted from the collaborative ltering dataset and summarises the e ect of the features on recommendation accuracy. The section also speci es how the features are used to form positive or negative evidence. Section 4 discusses the experiments performed, the experimental set-up and presents results. Conclusions are presented in Section 5.

2 Related Work

Collaborative ltering techniques produce recommendations for some active user using the ratings of other users, where these users have similar preferences to the active user. Collaborative ltering datasets can be predominantly distinguished by the fact that they are both large and sparse, i.e. in a typical domain, there are many users and many items but ratings only exist for a small percentage of all items in the dataset. The problem space can be viewed as a matrix consisting of the ratings given by each user for the items in a collection, i.e. the matrix consists of a set of ratings $r_{a,i}$, corresponding to the rating given by a user a to an item i. The problem space can equivalently be viewed as a graph where

nodes represent users and items, and nodes can be linked by weighted edges in various ways (e.g., user-item links; user-user links).

There has been much work undertaken in investigating weighting schemes for collaborative filtering where these weighting schemes typically try to model some underlying bias or feature of the dataset in order to improve prediction accuracy. For example, in [2] and [18] an inverse user frequency weighting was applied to all ratings where items that were rated frequently by many users were penalised by giving the items a lower weight. In [6] and [18] a variance weighting was used which increased the influence of items with high variance and decreased the influence of items with low variance. The idea of a *tf-idf* weighting scheme from information retrieval was used in [9] (using a row normalisation) and in [16] (using a probabilistic framework). More recent work in [3], [8] and [13] involve learning the optional weights to assign to items. In [11] a higher weighting is given to user neighbours who have provided good recommendations in the past (this weight is calculated using measures of "trust" for users) and in [4] items which are recommended more frequently are given a higher weighting (where the weight is calculated using an "attraction index" for items).

In general, although some of the weighting schemes for items have shown improved prediction accuracy (in particular those involving learning), it has proven difficult to leverage the feature information to consistently improve results. There may be a number of reasons for this including the fact that the dataset is sparse and also that the data may not always be correct. Even if the data is correct the underlying preferences that the data "describes" may not always be consistent as user tastes and opinions may change over time.

3 Methodology

In this paper, the focus is to extract implicit user and group information available from the collaborative filtering dataset to form a user model for each user and to use this model to provide evidence as to whether the system recommendations are accurate, or not, for a user. The implicit information extracted from the dataset is based on simple features which can be extracted from any recommendation dataset (e.g. number of items rated, average rating value) as well as extracting features which are based on measures from social network theory (degree, clustering coefficient) and from information retrieval (*tf-idf*).

3.1 Collaborative filtering approach

The collaborative filtering problem space is often viewed as a matrix consisting of the ratings given by each user for some of the items in a collection. Using this matrix, the aim of collaborative filtering is to predict the ratings of a particular user, a, for one or more items not previously rated by that user. Memory-based techniques are the most commonly used approach in collaborative filtering although numerous other approaches have been developed and used [2]. Generally, traditional memory-based collaborative filtering approaches contain three main stages (for some active user a):

– Find users who are similar to user a (the neighbours of a).
– Select the "nearest" neighbours of a, i.e. select the most similar set of users to user a.
– Recommend items that the nearest neighbours of a have rated highly and that have not been rated by a.

Standard statistical measures are often used to calculate the similarity between users in step 1 (e.g. Spearman correlation, Pearson correlation, etc.) [14]. In this work, similar users are found using the Pearson correlation coefficient formula.

3.2 User and group features

A user model is de ned which consists of six features. For some user a the features are de ned as follows:

– *rated* is the number of items rated by the user a.
– *avg-rating* is the average rating value given to items by the user a.
– *std-dev* is the standard deviation of the ratings of user a.
– *influence* is a measure of how in uential a user is in comparison to other users. As also considered in [13] and in [10], *influence* is de ned in this work by using measures from social network theory. In particular, the idea of degree centrality is used where the dataset is viewed as a graph (or social network) where nodes represent users and the values of weights on edges between users are based on the strength of similarity of users to each other (as shown in Fig. 1 with users linked if their Pearson correlation value is above 0.25). Degree centrality is then measured by counting the number of edges a node has to other nodes. Essentially this is a count of the number of neighbours (above a correlation threshold of 0.25) a user has.
– *clustering-coeff* is also a measure taken from social network theory and measures how similar users in a group are to each other using the clustering coefficient measure. This measures how connected the neighbours of the user a are to each other using the graph representation in Fig. 1. For example, if none of user a's neighbours are connected to each other, the clustering coefficient is 0, whereas if this sub-graph has a clustering coefficient of 1 then all of user a's neighbours are connected to each other.
  The clustering coefficient is calculated by dividing the number of actual links (*actual*) by the number of possible links between neighbour nodes for all neighbour nodes with degree greater than 1. Only user nodes that are connected to each other with a correlation value greater than 0.25 are considered neighbour nodes. For the collaborative ltering case, commonly used correlation measures are not commutative so therefore in the representation used, two edges can exist between two users. Therefore the total number of possible links that can exist between n nodes is $(n^2 - n)$.
  In addition, in the collaborative ltering case it is possible that small subgroups (small values of n) will have high clustering coefficients and therefore comparisons using clustering coefficient values may not always be meaningful. To overcome this the formula is extended to also include the active user

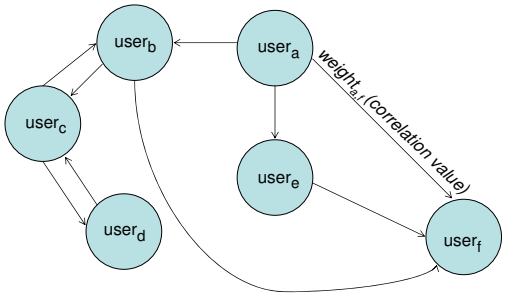


Fig. 1. Graph representation of users and their similarity.

in the calculation [17]. Thus the formula for the clustering coefficient for a user a with degree, $deg(a)$, and n neighbour nodes with degree greater than 1 becomes (1):

$$\frac{actual + deg(a)}{n(n+1)} \tag{1}$$

Considering the graph shown in Fig. 1 with the active user being $user_a$ who has three neighbours (b, e and f): user e is connected to user f and user b is connected to user f. Therefore the number of actual links is 2. The degree of the active node a is 3, therefore the clustering coefficient for this group is 0.42.

– *importance*: Some collaborative ltering weighting schemes incorporate the idea from Information Retrieval of a *term frequency, inverse document frequency* (*tf-idf*) weighting [9],[16]. The idea in information retrieval is to nd terms with high discriminating power, i.e. terms which "describe" the document well and also distinguish it from other documents in the collection. Mapping the idea of *tf-idf* to collaborative ltering, a "term" can be viewed as a user with associated ratings for M distinct items. The more ratings a user has the more important the user is, unless the items that the user has rated have been rated frequently in the dataset. Note that the value a user gives an item is not a frequency or a weight - it is an indication that the item has been rated and thus the actual rating value is not used in the following formula(2). The formula used to calculate the importance, w_i, of a user i is:

$$w_i = \frac{1}{M} \times \sum_{j=1}^{M} \left(1 + \log \frac{n}{n_j}\right) \tag{2}$$

where n is the total number of users in the dataset; M is the number of ratings by user i and n_j is the number of users who rated item j.

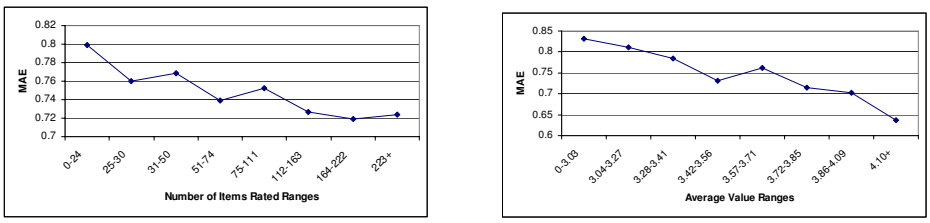
3.3 User and group features and their effect on accuracy

In this section the relative performance of a collaborative ltering approach is tested using di erent sets of users for each of the six features. A set of users

consists of the users who have the same value, or nearly the same value, for an identi ed feature. The aim is to ascertain which sets of users will be more likely to have better or worse predictions (measured using the mean absolute error (MAE) metric). For each feature, the range of values for that feature (e.g. [0,1] for the *std-dev* feature) is broken into regular intervals (typically 8 intervals) and users belong to a particular interval based on their value for that feature. All users in a particular interval then form a set. Intervals are chosen such that the set size (the number of users in each interval) is at least 100.

For testing, a standard subset of the Movie Lens dataset is considered. For each run, 30 users are chosen randomly from each set as the test users and 10% of their ratings for items are removed to yield the items to test (i.e. the system should return predictions for these items). In addition, for each feature a control set of 30 users is chosen randomly from the entire dataset as test users (i.e. the users are chosen without considering the feature value of these users). Results are averaged over 10 runs for each set of users, for each feature.

Fig. 2(a) shows the MAE results when the *rated* feature was analysed for eight sets of users. The *rated* value ranges from 0 to 668. The users in the rst set (0-24 interval) have rated 0-24 items; the users in the second set (25-30 interval) have rated 25-30 items; etc. The random group of 30 users (which are not included on the graph), with varying *rated* values, have an average MAE value of 0.7624 (over 10 runs). As expected, the worst MAE value for any set was for the users in the set who have rated between 0 and 24 items, i.e. these users have provided the very minimum number of ratings. Although we would expect that the accuracy should steadily increase as the number of ratings users have given increases, this was not necessarily the case. However, users who have rated close to the maximum number of items have the best MAE values.



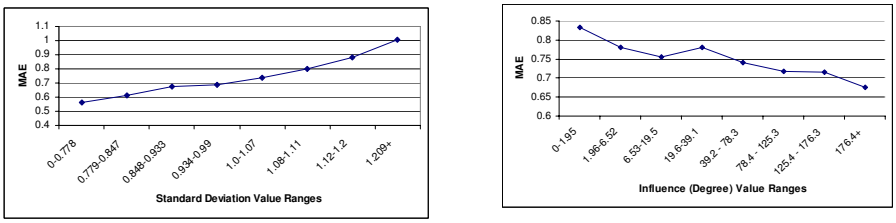
(a) *rated* (b) *avg-rating*

Fig. 2. *rated* and *avg-rating* MAE analyses.

Fig. 2(b) shows the MAE results when the *avg-rating* feature was analysed for eight sets of users. The MAE value for the randomly chosen users is 0.7321. The users with lowest averages (from the minimum to 3.03) have the worst MAE values and the users with the highest averages have the best MAE values.

Fig. 3(a) shows the MAE results when the standard deviation feature (*std-dev*) was analysed for eight sets of users. The *std-dev* value ranges from 0 to 1.718.

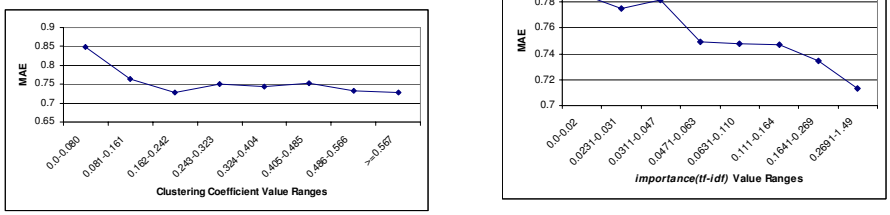
The users with low standard deviation (< 0.778) exhibited the best MAE values (average of 0.5595 in comparison to the MAE average of the randomly selected group which is 0.7779) while the users with the highest standard deviation had the worst MAE values. This suggests that better recommendations can be found for users with lower variance in their ratings.



(a) *std-dev* (b) *influence*

Fig. 3. *std-dev* and *influence* MAE analyses.

Fig. 3(b) shows the MAE results when the *influence* feature was analysed for eight sets of users. An $influence$ value of 0 means that a user has no neighbours. As expected, the users with fewest neighbours (0 or 1) have the worst MAE values and as the neighbourhood size grows there is a general trend towards lower MAE values. The average MAE value of the random group is 0.7508.



(a) *clustering-coeff* (b) *importance*

Fig. 4. *clustering-coeff* and *importance (tf-idf)* MAE analyses.

The clustering coefficient feature (*clustering-coeff*) was analysed for eight sets of users with values ranging from 0 to 0.864 where a value of 0 means that none of the active user's neighbours are linked to each other (with a correlation value above 0.25). As can be seen in Fig. 4(a), as the *clustering-coeff* value increases towards the maximum (i.e. the active user's neighbours are more similar to each other) the prediction accuracy very slightly improves. The poorest results are seen for users who have very low clustering coefficient values. The average MAE value of the random group is 0.7479.

The user importance feature (*tf-idf*) was also analysed for eight sets of users (see Fig. 3(b)). Results were poorer when a user has a low *tf-idf* weighting value and results are better when a user has a high *tf-idf* weighting value. The average MAE value of the random group is 0.725.

3.4 Forming the evidence

Based on the graphs in the previous section, thresholds were set for each of the features where, for each feature, a *evid-feature* parameter is set to 0 if the evidence is weak; *evid-feature* is set to 1 if the evidence is strong; and *evid-feature* is set to -1 otherwise, i.e., in this case it is not possible to clearly say whether this feature will have an e ect on recommendation accuracy. Therefore some users may receive item recommendations along with an indication that the recommendations have been formed with mostly weak evidence (where *evid-feature* is mostly 0) or mostly strong evidence (where *evid-feature* is mostly 1) or there may be no indication of strong or weak evidence if the *evid-feature* values are mostly -1 or there is a mixture of values for *evid-feature*.

4 Experiments and Results

The testing methodology involved checking if the evidence generated by the system, in terms of weak and strong evidence, is supported by the MAE results for users and per run, calculating:

- the percentage of users identi ed as having weak or strong evidence and the average MAE value of the users in both sets (the set of users identi ed as having weak evidence and the set of users identi ed as having strong evidence).
- the percentage of users not identi ed as having weak or strong evidence and the average MAE of these users.

We suggest that it is equally important to know when predictions have been formed using weak evidence as to know when predictions have been formed using stronger evidence. In order to test whether the system had correctly identi ed users with weak and strong evidence, the mean absolute error (MAE) metric was used to analyse results where the MAE value of each user was used to measure whether there is sufficient evidence in the dataset to form good recommendations. A user with an MAE value below the average MAE value is considered to have strong evidence and a user with an MAE value above the average MAE value is considered to have weaker evidence.

In the experiments performed, the Movie Lens dataset is used with 10% of users chosen randomly as test users and 10% of their items chosen randomly as test items. A nearest neighbour collaborative ltering approach using Pearson correlation is used to produce recommendations for all users and items in the test set. In addition, the system indicates whether weak or strong evidence exists for the user.

Fig. 5 summarises the results for 10 runs. On average, 32.87% of users (with average MAE of 0.86) were identi ed as having weak evidence. 37.77% of users (with average MAE of 0.61) were identi ed as having strong evidence. 29.36% of users (with average MAE of 0.78) in the test set were not identi ed as having weak or strong evidence. This shows that those users identi ed as having weak evidence have higher MAE values and thus are being given less accurate recommendations. Conversely, those users identi ed as having strong evidence have lower MAE values and thus are being given more accurate recommendations.

run	**%users identified with weak evidence**	avg. MAE (weak evidence)	**%users identified with strong evidence**	avg. MAE (strong evidence)	**%users not identified**	avg. MAE (not identified)
1	30.85	0.86	37.23	0.60	31.91	0.77
2	28.72	0.81	40.43	0.61	30.85	0.75
3	41.49	0.91	37.23	0.61	21.28	0.79
4	36.17	0.86	36.17	0.59	27.66	0.84
5	29.79	0.88	47.87	0.64	22.34	0.84
6	32.98	0.88	40.43	0.65	26.60	0.76
7	29.79	0.87	39.36	0.67	30.85	0.73
6	35.11	0.93	28.72	0.57	36.17	0.77
9	31.91	0.82	37.23	0.59	30.85	0.84
10	31.91	0.82	32.98	0.57	35.11	0.72
average	**32.87**	**0.86**	**37.77**	**0.61**	**29.36**	**0.78**

Fig. 5. Summary of results over 10 runs.

5 Conclusions and Future Work

In this paper the idea proposed is that a collaborative ltering system often has the information available to provide evidence as to whether the recommendations produced by the system are likely to be weakly or strongly supported. A user can thus be provided with more information with which to judge the recommendations which have been produced. The information used to obtain this evidence is already available in the collaborative ltering dataset and some of the information is calculated as part of the recommendation process. This information includes: the number of ratings given by a user, the average rating value given by a user, the standard deviation of user ratings, the number of neighbours a user has, the clustering coefficient value of a user and the importance of a user (using a *tf-idf* measure).

Results show that a large percentage of users are correctly identi ed as having weak or strong evidence. However further sample runs need to be performed and further analysis performed for the cases that were incorrectly identi ed. Future work will also consider the parameters and thresholds in more detail.

References

1. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
2. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative ltering. In *Uncertainty in Artificial Intelligence*, 1998.
3. K. Cheung and L.F. Tian. Learning user similarity and rating style for collaborative recommendation. *Information Retrieval*, 7:395–410, 2004.
4. A. deBruyn, L. Giles, and D.M. Pennock. O ering collaborative-like recommendations when data is sparse: The case of attraction-weighted information ltering. In *Adaptive hypermedia and adaptive web-based systems*, 2004.
5. T. George and S. Merugu. A scalable collaborative ltering framework based on co-clustering. In *Fifth International Conference on Data Mining*, 2005.
6. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative ltering. In *SIGIR*, pages 230–237, 1999.
7. R. Hu and Y. Lu. A hybrid user and item-based collaborative ltering with smoothing on sparse data. In *16th Intl. Conf. on Artificial Reality and Telexistence*, 2006.
8. R. Jin, J.Y. Chai, and L. Si. An automatic weighting scheme for collaborative ltering. In *SIGIR*, 2004.
9. G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, 2001.
10. B. Mirza, B. Keller, and N. Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20:131 – 160, March 2003.
11. J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 167–174, 2005.
12. M.P. O'Mahony, N. Hurley, and G.C.M. Silvestre. Detecting noise in recommender system databases. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, pages 109–115, 2006.
13. A.M. Rashid, G. Karypis, and J. Riedl. In uence in ratings-based recommender systems: An algorithm-independent approach. In *SIAM International Conference on Data Mining*, 2005.
14. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative ltering of netnews. In *Proceedings of ACM Conference on CSCW*, pages 175–186. Chapel Hill, 1994.
15. U. Shardanand and P. Maes. Social information ltering: Algorithms for automating word of mouth. In *CHI '95*, pages 210–217, 1995.
16. J. Wang, A. deVries, and M. Reinders. Unifying user-based and item-based collaborative ltering approaches by similarity fusion. In *SIGIR*, pages 501–508, 2006.
17. D.D. Wu and X. Hu. Mining and analyzing the topological structure of protein-protein interaction networks. In *Symposium on Applied Computing*, pages 185–189, 2006.
18. K. Yu, X. Xu, J. Tao, M.E. Kri, and H.-P. Kriegel. Feature weighting and instance selection for collaborative ltering: An information-theoretic approach. *Knowledge and Information Systems*, 5(2), 2003.

Evaluating Communication Strategies in a Multi Agent Information Retrieval System

David Lillis[1], Rem Collier[1], Fergus Toolan[2], and John Dunnion[1]

[1] School of Computer Science and Informatics
University College Dublin
{david.lillis, rem.collier, john.dunnion}@ucd.ie
[2] Faculty of Computing Science
Griffith College Dublin
fergus.toolan@gcd.ie

Abstract. With the complexity of computer systems increasing with time, the need for systems that are capable of managing themselves has become an important consideration in the Information Technology industry. In this paper, we discuss HOTAIR: a scalable, autonomic Multi-Agent Information Retrieval System. In particular, we focus on the incorporation of self-configuring and self-optimising features into the system. We investigate two alternative methods by which the system can configure itself in order to perform its task. We also discuss the Performance Management element, whose aim is to optimise system performance.

1 Introduction

Complexity is a serious issue with modern computer systems. As hardware has improved dramatically to facilitate the development of more and more powerful systems, the complexity of the software that runs on it has increased accordingly. As a result of this, the costs associated with administrating such systems has become a serious issue in the Information Technology industry.

This has resulted in a push towards the development of software systems that are capable of managing themselves, in order to reduce the input required from human administrators. Research in this area has come to be referred to as Autonomic Computing, a term coined by IBM in 2001 [1]. Essential features of an autonomic computing system include self-configuration, self-optimisation, self-protection, self-healing and a number of others [2–4].

In addition to these hardware and software changes, another feature of the computer industry in recent years has been the widespread adoption of the World Wide Web. This has resulted in a dramatic increase in the quantity of information being made available, through such things as news articles, blog entries and forum postings. As a result, Information Retrieval (IR) systems must be capable of dealing with more and more information, which has resulted the need for more sophisticated systems. Efficiency and scalability have been of increasing concern in the IR community, along with the presentation of high-quality results.

The development of IR systems using multi-agent techniques is not a new phenomenon [5–7]. Indeed, the aim of our work on HOTAIR (Highly Organised Team of Agents for Information Retrieval), a multi agent IR system with autonomic features [8], is not to demonstrate innovation in the development of multi-agent IR systems, but rather to investigate a range of techniques that will enhance the robustness and scalability of large-scale agent systems. Here, we focus on two approaches to agent interaction that facilitate self-configuration.

Accordingly, Section 2 outlines reasons why IR is an ideal testbed for an Autonomic Computing System. Section 3 presents a general overview of the HOTAIR architecture, of which two versions have been developed. The first incorporates a Broker agent which maintains a centralised view of the entire system and manages the behaviour of other agents. An alternative architecture allows individual agents more autonomy by allowing them to gather more knowledge about their environment. Each version features a Performance Manager to provide elements of self-optimisation. Section 4 outlines experiments carried out to compare the performance of these two architectures. Finally, we include our conclusions and ideas for future work in Section 5.

2 Information Retrieval as an Autonomic Computing Testbed

An IR system is an ideal testbed for autonomic computing for a number of reasons. Firstly, a number of essential features of autonomic computing systems that have great relevance to IR systems:

- **Self-Configuration:** As more and more information becomes available to be processed by IR systems, it becomes necessary for the systems themselves to grow accordingly. This necessitates the introduction of additional hardware into the system, to cope with the increased workload. As it would be unfeasible to merely transfer the data from one system to another, it is necessary to add this hardware while the system is running. A self-configuring system would be capable of incorporating these additional resources into the system and make use of them, without intervention by a human administrator.
- **Self-Optimisation:** As with any large-scale system, optimal use of the available resources is an important aim. A self-optimising system will monitor its own use of resources and can react immediately to exploit any opportunities for greater performance that may arise.
- **Self-Healing and Self-Protection:** Recovery from failures and protection from external attack are essential to ensure reliability in any large-scale system, including an IR system. Given the huge revenues that are earned in advertising alongside, for example, online searches, prolonged downtime could have disastrous consequences for popular IR systems.

In addition to the above, the IR domain is an attractive one in which to carry out research on autonomic computing. The barriers to entry are low, as an IR system can be run on a cluster of low-end desktop computers. Also, the

existence of open source IR libraries such as Lucene[3] and Xapian[4] mean that the learning curve in setting up an IR system is relatively shallow in comparison to some other domains.

3 The HOTAIR Architecture

HOTAIR is an Information Retrieval system developed as a Multi-Agent System. It is written in the AFAPL2 agent programming language [9] and runs within the Agent Factory framework, a FIPA-compliant runtime environment for agents [10, 11]. The aim of the HOTAIR project is the development of a reliable and scalable agent-based search engine architecture. As discussed in Section 2, this application domain was chosen because we believe that IR is a problem that offers significant challenges in terms of the scale of the applications, which invites the use of autonomic features. Consistent with a typical IR system, HOTAIR consists of two distinct subsystems:

The Indexing Subsystem is reponsible for identifying new documents to make available to users, whether from the World Wide Web, an FTP site, a ZIP archive, a file share or some other source. These documents must be added to a searchable index from which results will be extracted to be presented to users.

The Querying Subsystem is responsible for accepting queries from users, running those queries against the index built up by the Indexing Subsystem and returning those results to the user.

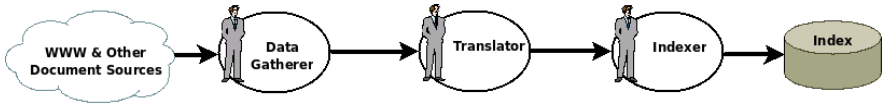


Fig. 1. Flow of documents through the HOTAIR Indexing System

The focus of this paper is on the Indexing Subsystem. In order for a document to be added to the index, it must go through three stages, illustrated in Figure 1. Each of these three steps is carried out by what is referred to as a "core agent". *DataGatherer* agents are responsible for identifying and collecting new documents for inclusion in the index. These documents may be taken from any number of sources and may be in a variety of file formats. *Translators* are necessary to interpret the documents of various file types that have been collected by the DataGatherers. These are converted into a common file type, known as the HOTAIR Document Format (HDF), which maintains an XML representation of the document contents. *Indexers* represent the final step that a document must go through in order to be included in the system's index. Indexer agents accept

[3] http://lucene.apache.org
[4] http://www.xapian.org

HDF documents as their inputs, extract the contents from these documents and save this data in the index. In order to reduce the amount of communication necessary between agents, documents are not processed individually, but rather in bundles of 20. We refer to these bundles as "jobs".

Initially, core agents are not aware of the location of other core agents. The discovery of, and interaction with, other relevant core agents is an element of self-configuration which we aim to introduce in the following sections. During the development process, two versions of the HOTAIR architecture were created. The principal difference between these is the way in which agents gain knowledge about their environment and discover other agents with which they must interact in order to carry out their tasks. Section 3.1 describes the basic workflow of the system, which both architectures share. Of the two architectures, the "broker architecture" uses a Broker agent to micro-manage the behaviour of the core agents. This is presented in Section 3.2. Section 3.3 presents the alternative "broadcast architecture", in which the core agents have more control over their own behaviour, as they have access to more information about the state of other agents and of the system as a whole. Finally, section 3.4 describes the performance management features that are applicable to both architectures.

3.1 Indexing System Workflow

This section describes the workflow of the HOTAIR Indexing Subsystem: how the core agents interact with one another in order to include documents in the index. Activities discussed in this section are carried out by core agents in both the brokered and broadcast architectures. We use the term "Processor" to describe any agent that receives jobs from another agent and processes them in some way (i.e. Translators process jobs taken from DataGatherers and Indexers process jobs taken from Translators). "Provider" is a generic term to refer to any agent that provides jobs for a Processor (i.e. DataGatherers provide jobs for Translators, which in turn provide jobs for Indexers).

At any given time, each Processor is assigned to a single Provider, from which it receives documents for processing. The way in which this assignment takes place is the fundamental difference between the brokered and broadcast architectures and is discussed in detail in the following sections.

The actual flow of jobs through the system then follows a pull-style pattern. When a Processor has the capacity to process a job, it requests a job from the Provider to which it is assigned. Each Provider maintains an output queue, which contains jobs on which it has completed its own processing. It is from this queue that a job is taken and forwarded to the Processor that requested it. If the Provider's output queue is empty, it will reply with that information, at which time the Processor must be reassigned to an alternative Provider. The Performance Manager (discussed in Section 3.4) is charged with ensuring that the system is balanced so there are always documents available for processing.

Once the Processor has finished processing the job, it adds the job to its output queue if it is also a Provider. Indexers do not have output queues, as they represent the final stage of processing a document must undergo.

3.2 Brokered Architecture

The Brokered version of HOTAIR employs a Broker agent to aid the core agents in the discovery of the other core agents with which they need to communicate and interact. Brokers have long been seen as a useful design pattern for creating more open agent systems [12, 13]. Here, the core agents have no first-hand knowledge of their environment other than the location of the Broker agent.

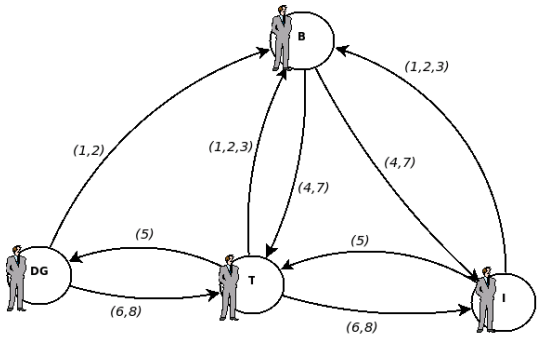


Fig. 2. Communication in the Brokered Architecture

Figure 2 illustrates the communications between agents in the Brokered Architecture. For simplicity, only one agent of each type is shown, though many DataGatherers, Translators and Indexers may exist in practice. Upon creation, each new core agent must register with the Broker (shown as communication *(1)* in Figure 2) and is included in the model the Broker uses to assign Processors to Providers. Each Provider periodically contacts the Broker to inform it of the number of jobs currently contained in its output queue *(2)*. This allows the Broker to build a model of the amount of work being created for each category of Processor.

In order to find jobs to process, a Processor must contact the Broker and request that it be assigned to a Provider *(3)*. The Broker then makes use of its knowledge of the output queue status of the relevant Providers and of the assignments it has already made to make the most appropriate assignment *(4)*. A Processor will continue requesting jobs from the same Provider *(5,6)* until either the Broker reassigns it to a different Provider *(7)* or the Provider informs it that its output queue is empty *(8)*. In the latter case, the Processor must re-contact the Broker to be reassigned to an alternative Provider.

A key advantage of such an architecture is that all agent assignments are made with full knowledge of the state of the system as a whole. Thus, the allocation of agents can at all times be balanced so as to ensure that all Providers will eventually have Processors assigned to them. Assuming the Broker is using an appropriate assignment algorithm, this has the effect that jobs cannot become

held up in a situation where they are located in the output queue of a Provider that never has a Processor assigned to it.

This form of Broker agent does, however, introduce a single point of failure into the system. If Broker fails or becomes uncontactable, the system as a whole will also fail. Core agents will no longer be able to identify Providers, as they will not have a method of finding their location. As the Broker is also the sole agent that is aware of the type, location and status of the core agents, this information is also lost if the Broker fails. In order to overcome this limitation, work has been carried out on the development of robust brokered architectures in order to maintain system performance in the event of the failure of the Broker [14].

3.3 Broadcast Architecture

The second version of the HOTAIR architecture involved Providers broadcasting the state of their output queues to all other core agents, rather than just to a centralised Broker agent. Specifically, a wild card ("*") was introduced into the agent name component of the FIPA agent identifiers allowing the partial specification of receiver agents. An example of this would be the agent identifier $agentID(ind*, addresses(http://localhost:444/acc))$[5], which could be used to send a message to all agents on the specified agent platform whose name beings with "ind" (which, in conjunction with an appropriate naming scheme, could be used to contact all Indexer agents). Furthermore, replacing "ind*" with "*" is akin to a broadcast to all agents on the specified agent platform. Following this, a UDP multicast Message Transport Service was introduced, which, together with the wild card, allowed broadcasting of message to all agents on all platforms that are listening via the relevant UDP agent communication channel.

By introducing the above message broadcasting mechanism, and allowing Providers to broadcast their state, the need for the Broker agent is removed. Each core agent can build its own model of its environment and decide for itself which Provider from which it will request jobs. Perhaps the key issue in employing a UDP-based agent communication channel is that agent communication is no longer guaranteed. However, so long as the approach is used judiciously (e.g. repeated broadcasting of status updates) and in conjunction with guaranteed communication mechanisms (e.g. for job requests) the overall behaviour of the system can be guaranteed without affecting the robustness of the system. For some tasks, however, it would be useful to have the ability to broadcast reliably, and so we intend to investigate alternative methods of broadcasting using XMPP-based technologies such as Jabber[6].

The broadcasts made by core agents include information about the length of their output queue, the number of documents in jobs processed and the time taken to perform this processing. In addition to being used by other core agents to support the system workflow, it is also possible for a Performance Manager to make use of this information to influence its management decisions.

[5] This identifier is specified in the format that is employed by the AFAPL programming language

[6] http://www.jabber.org

Using this broadcast architecture, assignment of Processors to Providers is carried out by the Processors themselves. By default, agents assign themselves to the Provider that has most recently broadcasted the largest output queue. This self-assignment allows the agents to maintain the functioning of the system even in the absence of any management agents, thus removing the single point of failure without the need to introduce additional strategies to improve the robustness of the Broker. This self-assignment can, however, be overridden by instructions from a Performance Manager agent to assign it to another Provider. Once a Processor has assigned itself to a provider, it requests jobs in the same way as in the Brokered Architecture. On receipt of a message to inform it that its Provider no longer has any jobs available for processing, a Processor will reassign itself to another Provider.

3.4 Performance Management

Self-optimisation of the system is carried out by a centralised Performance Manager. The aim of this manager is to maximise throughput. In order to carry out this task, it has a number of actions available to it so as to alter system behaviour.

Agent Reassignment: In the broadcast architecture, agents will by default assign themselves the Provider with the greatest number of outstanding jobs. However, this does not necessarily lead to optimal system performance as it is not optimised for the system as a whole. The Performance Manager may override this default behaviour so as to improve system efficiency. In the brokered architecture, all agent assignments are performed by the Broker, which doubles as the Performance Manager.

Group Halting/Resumption: Whenever a Provider group processes jobs quicker than the agents assigned to it, the Performance Manager may instruct members of that group to temporarily cease processing jobs, so as to allow the backlog to be cleared.

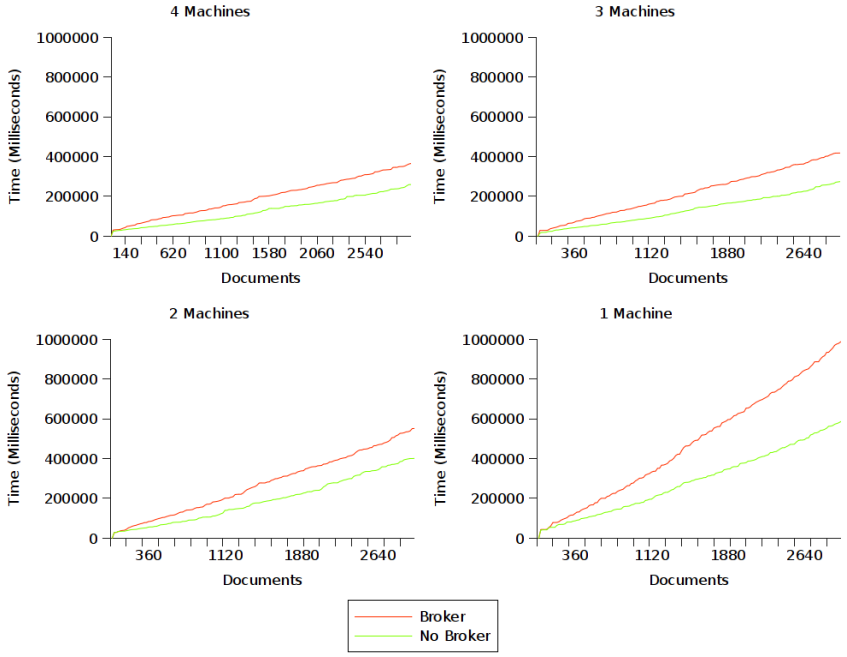
Agent Creation: Initially, the system begins with a small agent community. Once the system is running, the Performance Manager incrementally increases the size of the agent community by creating agents for the group it believes will benefit most by the creation of an additional agent.

Agent Destruction: When the system reaches its maximum capacity (i.e. no more agents can be created), the Performance Manager investigates whether it is possible to create free space for new agents by destroying unnecessary agents, or agents from overpopulated groups.

4 Experiments and Evaluation

In order to compare the performance of the brokered and broadcast architecture, a number of experiments were run. Each time the system was run, three Data-Gatherers were created, each gathering documents from a different document

Fig. 3. Document Indexing Speed



collection. The collections used were Cranfield, NPL and the 2Gb Web Track collection from the TREC conference.

The number of machines available to the system was increased so as to evaluate the scalability of each of the two architectures. Each machine is a standard desktop computer running an Agent Factory agent platform, on which the agents run. The creation of agents on any of these platforms was the decision of the Performance Manager, which was present for both architectures. In the brokered architecture, the Performance Manager also took on the the role of a Broker. The maximum number of agents that could be created on each platform was manually set at 15. Systems were evaluated by their performance in successfully indexing 3,000 documents. As the document collections contain more than 3,000 documents, in each case all elements of the system were still running on the termination of the experiment. The results of running these experiments on between 1 and 4 machines are displayed in Table 1 and Figure 3. In each case, the system was run three times and all values used are the average of these three runs. Figure 3 shows the number of documents indexed by each system configuration over time. Table 1 shows the overall time taken to complete the indexing of the 3,000 documents.

For each configuration of the system, the broadcast architecture outperformed the brokered architecture to a large degree. The extent of this difference

Table 1. Time to index 3,000 documents (in milliseconds)

	Broker	Broadcast	% difference
1 machine	987849	585957	-40.68%
2 machines	551573	400895	-27.32%
3 machines	418200	274210	-34.43%
4 machines	365612	259286	-29.08%

was always in excess of 27%. The addition of an extra machine reduced the total processing time in each case, as expected. An interesting feature to note is that the impact of the addition of the fourth machine was not as dramatic as that of the second or third machines. This would suggest that the Performance Manager is not currently making full use of the resources being made available to it. Although this is an interesting observation, it does not affect the comparison between the brokered and broadcast architectures, as the same performance management strategy was used in both.

5 Conclusions and Future Work

This paper evaluates two approaches to agent interaction that have been employed in the HOTAIR architecture. The first approach, which is presented in Section 3.2, is based on best practices and employs a Broker as a middle agent, which manages interactions between the underlying core agents (DataGatherers, Translators and Indexers) through the maintenance of a model of the current state of the system. In contrast, the second approach, which is presented in Section 3.3, removes the need for a broker through the introduction of a UDP-based broadcast mechanism and the incorporation of a partial system model internally within each core agent.

Central to both architectures is the Performance Manager, builds an extended model of the system state that includes location and assignment information. The Performance Manager performs periodic analysis of the current system state, modifying the assignments between core agents in order to improve the performance of the system. It is also empowered with a number of other actions it can take to optimise system performance.

As is shown through the results presented in Section 4, the broadcast architecture consistently outperforms the broker architecture by more than 27%. This is in addition to the advantages it provides in terms of robustness.

Future work will involve further investigation on the scalability of the system on larger clusters of machines. Additionally, we aim to focus on the development of appropriate self-optimisation algorithms, which will make best use of the available performance management actions (outlined in Section 3.4) so as to increase system throughput. Another key element of autonomic computing which we aim to address is self-healing, which should ensure that the system should be

capable of tolerating and recovering from agent failures, while ensuring that all documents are successfully indexed.

References

1. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology. Manifesto, IBM Research (October 2001)
2. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. IBM Systems Journal **42**(1) (2003) 5–18
3. Sterritt, R., Bustard, D.W.: Autonomic computing - A means of achieving dependability? In: ECBS, IEEE Computer Society (2003) 247–251
4. Hanson, J.E., Whalley, I., Chess, D.M., Kephart, J.O.: An architectural approach to autonomic computing. In: ICAC '04: Proceedings of the First International Conference on Autonomic Computing (ICAC'04), Washington, DC, USA, IEEE Computer Society (2004) 2–9
5. Lazarou, V., Clark, K.: A multi-agent system for distributed information retrieval on the world wide web (1997)
6. McDermott, P., O'Riordan, C.: A system for multi-agent information retrieval. In: AICS '02: Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science, London, UK, Springer-Verlag (2002) 70–77
7. Odubiyi, J.B., Kocur, D.J., Weinstein, S.M., Wakim, N., Srivastava, S., Gokey, C., Graham, J.: Saire-a scalable agent-based information retrieval engine. In: AGENTS '97: Proceedings of the first international conference on Autonomous agents, New York, NY, USA, ACM Press (1997) 292–299
8. Peng, L., Collier, R., Mur, A., Lillis, D., Toolan, F., Dunnion, J.: A self-configuring agent-based document indexing system. In: Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2005), Budapest, Hungary, Springer-Verlag GmbH (2005)
9. Ross, R., Collier, R., O'Hare, G.: Af-apl bridging principles & practice in agent oriented languages. In: Proceedings of the First International Workshop on Programming Multiagent Systems, Languages and Tools - PROMAS 2004, New York, USA (2004)
10. Collier, R., O'Hare, G., Lowen, T., Rooney, C.: Beyond prototyping in the factory of agents. In: Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003), Prague, Czech Republic, Springer-Verlag GmbH (2003) 383
11. Collier, R.: Agent Factory: A Framework for the Engineering of Agent-Oriented Applications. PhD thesis, University College Dublin (2001)
12. Decker, K., Sycara, K., Williamson, M.: Middle-agents for the internet. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan (1997)
13. Kolp, M., Do, T.T., Faulkner, S., Hoang, T.H.: Introspecting Agent-Oriented Design Patterns. In: Handbook of Software Engineering and Knowledge Engineering. Volume 3: Recent Advances. World Scientific (2005) 105–134
14. Kumar, S., Cohen, P.R.: Towards a fault-tolerant multi-agent system architecture. In: AGENTS '00: Proceedings of the fourth international conference on Autonomous agents, New York, NY, USA, ACM Press (2000) 459–466 Reference for robust Broker-based system.

One-Class Support Vector Machine Calibration Using Particle Swarm Optimisation

Yang Liu, Michael G. Madden
Department of Information Technology,
National University of Ireland, Galway, Ireland
sharkyangliu@yahoo.co.uk; michael.madden@nuigalway.ie

Abstract. Population-based search methods such as evolutionary algorithms, shuffled complex algorithms, simulated annealing and ant colony search are increasingly used as automatic calibration methods for a wide range of numerical models. This paper proposes the use of particle swarm optimisation to calibrate the parameters a one-class support vector machine. This approach is developed and tested in the calibration of a one-class SVM, applied to several data sets. The results indicate that the proposed method is able to match or surpass the performance of a one-class SVM with parameters optimized using a standard grid search method, with much lower CPU time required.

1 Introduction

During the past two decades, a great deal of research has been devoted to the development of traditional classification methods (binary and multi-class classification) for pattern recognition [1]. Such research has focused primarily on four issues: (1) Determination of appropriate quantity and most informative kind of data (feature selection); (2) Dimension reduction for high dimensional features (feature selection); (3) Search for a classifier that can reliably solve linear or non-linear classification problems; and (4) Validation for the classifier. Non-linear classification methods such as Support Vector Machines (SVMs), K-Nearest Neighbour (KNN) classifier, Decision Tree and Artificial Neural Networks (ANNs) are widely used traditional classification problems.

However, for a significant number of practical problems, traditional *discriminating* classifiers that are trained using positive and negative examples are not directly applicable, because negative examples may be either rare, entirely unavailable or statistically unrepresentative. Such problems include industrial process control, text classification and image analysis. One-Class Classification (OCC) is emerging as a solution, which characterizes the target class, seeking to distinguish one class from the universal set of multiple classes.

One class classification (OCC) algorithms are receiving increasing interest both in the academia and industry [2, 3]. In some real-world applications, negative examples are hard or expensive to collect and label. Either a negative doesn't exist, or collection and label of the negative is computationally very expensive. In an example of diagnosis of a disease, positive data are easy to access (e.g., all patients who have disease) and unlabeled data are abundant (e.g., all patient), but negative data are expensive if detection tests for the disease are expensive since all patients in the

database cannot be assumed to be negative samples if they have never been tested. The second example is system intrusion data. Historical data of system intrusion cannot be used to recognise new kinds of assault. An effective security tool would be one designed to recognise assaults as they occur through the understanding and comparison of the current behaviour against nominal systems activity. Another example is the tool break detection problem. The challenge of the tool break detection problem lies in the break data is relative scarcity compared with normal cutting data since it is difficult and costly to obtain especially for new tool types and cutting tasks. In both cases, it is necessary to estimate the test example by constructing a new classifier which does not depend on such negative examples would be especially desirable.

The One-Class Support Vector Machine is a general purpose learning method designed to handle the various one-class classification problems [4, 5]. The algorithm maps the data into the feature space corresponding to the kernel, with the outliers mapped to a small region enclosing the origin, and target class instances are separated from the origin with maximum margin. For a new point x, the value $f(x)$ is determined by evaluating which side of the hyperplane it falls on, in feature space.

To turn the one class SVM algorithm into an easy-to-use black-box method for practitioners, questions about the selection of parameters (such as the width of a Gaussian kernel, and the upper bound on the fraction of training errors and the lower bound of the fraction of support vectors ν) must to be tackled [4, 5]. The method proposed in this paper investigates the use of automatic calibration of one-class SVM to find the optimal parameters. Calibration is the process of modifying the input parameters to a numerical model until the output from the model matches an observed set of data [6]. In automatic calibration, parameters are adjusted automatically according to a specified search scheme and numerical measures of the goodness-of-fit. Compared to manual calibration, automatic calibration is faster while being less dependent on individual skill and effort, and relatively easy to implement. Previous work has involved the development and application of optimization algorithms for automatic model calibration, with the proposed methodology being demonstrated on numerical model calibration applications [6].

This paper proposes a novel approach that combines particle swarm optimization (PSO) with one-class SVM, called the PSO-One-Class SVM (POCS) hybrid algorithm. The proposed methodology is demonstrated on several applications, showing that the proposed POCS method possesses better ability to find good parameters (λ and ν) using one-class SVM in some applications. Performance comparison between PSO and a basic grid search approach is then presented.

2 One-Class Support Vector Machine

Schölkopf et al. have proposed a strategy of mapping the target-class data into the feature space corresponding to the kernel and to separate them from the origin using a boundary with maximum margin [4, 5]. The algorithm is an extension of the binary support vector algorithm to the case of one-class data. As described by Manevitz and Yousef [3], it is supposed that there is a dataset drawn from an underlying probability

distribution *P*, and one needs to estimate a "simple" subset *S* of the input space such that the probability that a test point from *P* lies outside of *S* is bounded by some a prior specified as $v \in (0,1)$. The solution for this problem is obtained by estimating a function *f* which is positive on *S* and negative on the complement S^C [4, 5]. This is illustrated in Figure 1 in which target class data are labelled as +1 and outliers are labelled as -1. The origin is the only original point that is not a member of the target class, but the algorithm relaxes this constraint to return a function *f* that has the value -1 in a restricted region around the origin and +1 elsewhere.

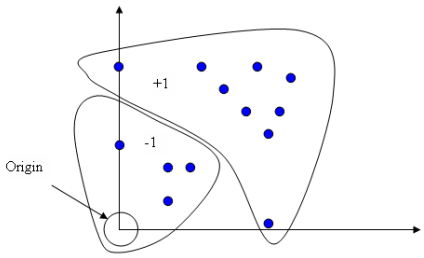


Figure 1: One-class SVM Classifier (Source: Manevitz and Yousef [3]).

To separate the data from the origin, we solve the following quadratic program:

$$\min \frac{1}{2}\|\omega\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\xi_i - \rho \tag{1}$$

subject to

$$(\omega \bullet \Phi(x)) \geq \rho - \xi_i \qquad i = 1,2,\ldots,l \quad \xi_i \geq 0 \tag{2}$$

where ω and ρ are hyperplane parameters, Φ is the map from input space to feature space, v is the asymptotic fraction of outliers allowed, l is the number of training instances, and ξ is a slack variable. For solutions to this problem, ω and ρ, the decision function

$$f(x) = sign((\omega \bullet \Phi(x)) - \rho) \tag{3}$$

specifies labels for test examples, e.g., -1 for outliers.

Two commonly used kernel functions are the Gaussian Radial Basis Function (RBF) kernel $k(x_1, x_2) = \exp(-\lambda\|x - y\|^2)$ and the polynomial kernel $k(x_1, x_2) = (x \cdot y + c)^d$, where the free parameter *d* is the degree of the polynomial kernel.

3 Particle Swarm Optimisation

Kennedy and Eberhart developed particle swarm optimisation based on the analogy of swarming animals, such as a flock of birds or school of fish [7]. . In each iteration,

each agent is updated with reference to two "best" values: *pbest* is the best solution (in terms of fitness) the individual particle has achieved so far, while *gbest* is the best obtained globally so far by any particle in the population. Each agent seeks to modify its position using the current positions, the current velocities, the distance between the current position and *pbest*, and the distance between the current position and *gbest.*

The velocity of each agent is modified by the following equation:

$$v_i^{k+1} = K[v_i^k + c_1 \times rand() \times (pbest_i - s_i^k) + c_2 \times rand() \times (gbest - s_i^k)] \quad (4)$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{where} \quad \varphi = c_1 + c_2 \qquad \varphi > 4 \quad (5)$$

A modification, the *constriction factor* approach, can generate higher quality solutions than the conventional PSO approach [8]. Here, the current position can be modified by the following equation:

$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad (6)$$

Compared to genetic algorithm optimisation, there are not many parameters that need to be tuned in PSO. The parameters are: the number of particles; weighting factors; and the maximum change for a particle. It is generally found that operation is not very sensitive to parameter settings. For the number of particles, the typical range is 20 – 40 [9]. The weighting factors, c_1 and c_2, are often to 2 [9], though other settings are used in different papers, typically with $c_1 = c_2$ and in the range [0, 4] [9].

4 Automatic Calibration Scheme

The general flow chart for the calibration process using PSO is presented below and illustrated in Figure 2:

Step 1: Generation of initial condition of each agent
We begin with initial population and velocities sampled randomly from the feasible space.

Step 2: Evaluation of search point for each agent
The objective function value is calculated by running one-class SVM model.

Step 3: Modification of each searching point
The current searching point of each agent is changed using (4) and (5). If the value is better than the current *pbest* of the agent, *pbest* is replaced by the current value. If the best value of *pbest* is better than the current *gbest*, *gbest* is replaced by the best value.

Step 4: Stop.
As the standard PSO's search progresses, the entire population tends to converge towards the global optimum. This process is continued until a satisfactory condition is

met. The termination criterion is determined according to whether the maximum number of generations or a designated value of fitness is reached.

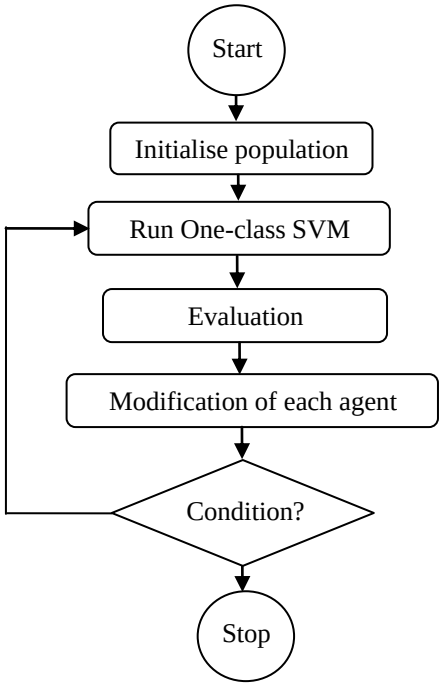


Figure 2: Outline of PSO for Optimisation Problems

5 Evaluation

5.1 Fitness Function

In order to obtain successful calibration by using automatic optimisation routines, it is necessary to formulate the calibration objective. The fitness function is formulated as follows (minimisation of fitness is assumed):

$$f(\lambda,\nu) = \frac{1}{1+PositiveRate} \tag{7}$$

$$PositiveRate = \frac{\text{Targets correctly classified}}{\text{Total targets in test set}} \tag{8}$$

For maximization problems, the fitness can be calculated as the reciprocal of the objective function value so that solutions with larger objective function value get smaller fitness.

5.2 Datasets

In order to test the validity of the proposed methodologies, the POCS method was applied to the following datasets:

Tremor Dataset: (The Tremor dataset from Exeter University, UK)[1].
For the calibration, 89 positive examples and 90 negative examples were used for training. In order to evaluate the performance of the calibrated models, test data (45 positive examples and 46 negative examples) and validation data (44 positive examples and 43 negative examples) were used.

Diabetes dataset (The Pima Indians Diabetes Database from UCI) [2] .
For the calibration, 185 positive examples and 327 negative examples were used for training. In order to evaluate the performance of the calibrated models, test data (34 positive examples and 95 negative examples) and validation data (49 positive examples and 78 negative examples) were used.

Vehicle dataset (The vehicle dataset from Statlog, to recognize a vehicle from its silhouette). [2] For the calibration, 219 positive examples and 63 negative examples were used for training. In order to evaluate the performance of the calibrated models, test data (202 positive examples and 80 negative examples) and validation data (213 positive examples and 69 negative examples) were used.

When negative examples (objects which should be rejected) are available, they can be used during the training to improve the performance [2].

5.3 Experiment Setup

In our research we used the OSU SVM (version 3.0). The OSU SVM Support Vector Machine Toolbox for MATLAB uses the LIBSVM packag[3]. The relevant experiment parameters using the PSO for one-class SVM calibration are listed in Table 1.

Table 1: Experimental Parameters

Parameter	Description	Range
λ	Kernel parameter	[0.0001 100]
ν	Fraction of outliers and support vectors	[0.01 0.3]
C_1	Weighting factor 1	2.5
C_2	Weighting factor 2	2.5
G	The total iterations	30
P	The number of particles	60

[1] The dataset is available at http://www.dcs.ex.ac.uk/studyRes.
[2] The dataset is available at http://www-it.et.tudelft.nl/~davidt/occ/index.html.
[3] OSU SVM 3.0 is available at http://svm.sourceforge.net/download.shtml.

The results, including the optimal parameters, positive rate, best and worst calibration results, average values, and the standard deviations using PSO for the objective function after 30 generations with a population size 60, are listed in Table 2. Table 2 also shows the validation result results of applying the calibrated parameter set to the validation dataset. From Table 2, it can be seen that PSO is able to find optimal calibration parameters of one-class SVM with good positive rates of the 10 random runs, and all the negative examples can be classified (negative rate =100%) in the ten random runs. The negative rate is formulated as follows:

$$NegativeRate = \frac{\text{Outliers correctly classified}}{\text{Total outliers in test set}} \tag{9}$$

The small standard deviations of fitness by the PSO imply that the method POCS is stable.

Table 2: One-Class SVM Calibration and Validation Results Using PSO

Calibration and validation results using PSO for the tremor dataset				
Trial	Optimal Parameter λ	Optimal Parameter ν	Calibration Result Positive Rate (%)	Validation Result Positive Rate (%)
Best	0.0054	0.02	100	88.64
Worst	0.1371	0.01	91.11	88.64
Mean			97.111	87.048
STD			3.4831	2.6368
Calibration and validation results using PSO for the diabetes dataset				
Trial	Optimal Parameter λ	Optimal Parameter ν	Calibration Result Positive Rate (%)	Validation Result Positive Rate (%)
Best	0.0001	0.03	100	100
Worst	0.0001	0.04	100	100
Mean			100	100
STD			0	0
Calibration and validation results using PSO for the vehicle dataset				
Trial	Optimal Parameter λ	Optimal Parameter ν	Calibration Result Positive Rate (%)	Validation Result Positive Rate (%)
Best	4.0090	0.01	99.50	99.06
Worst	0.0285	0.03	99.01	98.12
Mean			99.402	98.872
STD			0.2066	0.3963

Figure 3 shows an example that entire population converged around the global optimum after 3 generations using PSO with the population size of 60 for the tremor dataset, so a fixed number of iterations or generations (G=30) has been suggested as a stopping criterion in the calibration process for the three test datasets.

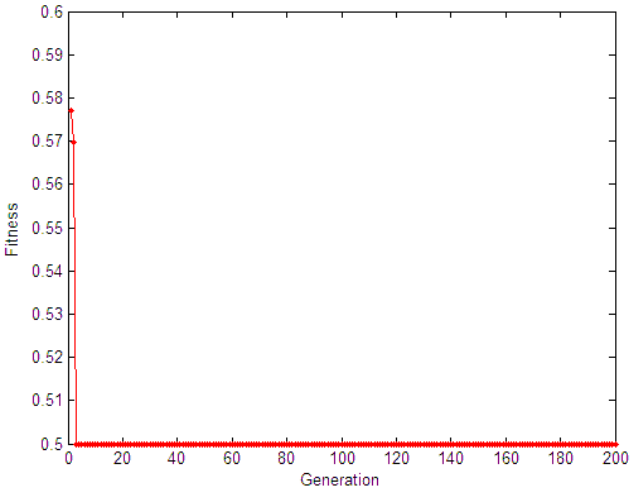


Figure 3: Iteration process using PSO

5.4 Comparison with Grid Search

For the purposes of comparison with PSO, we perform an standard grid search over the same range of parameters and with the same increments; i.e. $\lambda = 0.0001, 0.1001,...,$ 100 and $\nu = 0.01,0.03,\ldots, 0.3$. Thus, for the grid search, the number of model evaluations equal to 15×1000. The grid search method is simply an exhaustive search to determine the global optimum among those at each point on the grid of parameter values. Clearly, it is not very efficient, but is deterministic and reliable. If *n* is the number of parameters, the method employs a moving *n*-dimensional grid with spacing determined by the increment specified. The algorithm tries to centre the grid around the minimum point for each dimension (parameter), moving in an appropriate direction during each iteration. The optimization is successful when the grid becomes centered on a minimum point across all dimensions. Table 3 also shows the results of using the basic grid search to find that the best (λ, ν) and calibration and validation results.

From Tables 2 to 3, it is seen that the results of PSO and grid search are very close. Thus, it is clear that the PSO optimisation framework considered here is capable of searching more efficiently than the standard grid method on the objective function under a limited computational budget (60×30 evaluations). The above results indicate the number of function evaluations using PSO is around 88 percent less than grid search. This implies that we get a considerable advantage by using PSO.

Table 3: One-class SVM Calibration and Validation Results Using grid search

Calibration and validation results using grid search method for the tremor data				
Evaluations	Optimal	Optimal	Calibration Result	Validation Result

	Parameter λ	Parameter ν	Positive Rate (%)	Positive Rate (%)
15×1000	0.0001	0.2	100	88.64

Calibration and validation results using grid search method for the diabetes data				
Evaluations	Optimal Parameter λ	Optimal Parameter ν	Calibration Result Positive Rate (%)	Validation Result Positive Rate (%)
15×1000	0.1001	0.1	85.29	85.71

Calibration and validation results using grid search method for the vehicle dataset				
Evaluations	Optimal Parameter λ	Optimal Parameter ν	Calibration Result Positive Rate (%)	Validation Result Positive Rate (%)
15×1000	3.7001	0.01	99.06	99.50

6 Concluding Remarks

When using one-class SVM for classification problems, it is difficult to decide the width of a Gaussian kernel λ, and the upper bound on the fraction of training errors and the lower bound of the fraction of support vectors ν. To tackle this problem, an automatic calibration scheme has been formulated that considers the calibration problem in a general single objective framework. The scheme seeks to optimise the true positive rate. The hybrid method POCS was presented that can find good parameters for a one-class SVM. It has been shown that the proposed method performed more efficiently when compared with traditional grid search method. The simulation results indicated that the proposed method was able to reduce the required simulation runs to 12% of grid search while achieving comparable calibration and validation results. The results provide us with confidence that the proposed method is indeed a viable method to reduce the computation effort required in calibrating one-class SVM model.

Acknowledgments. The authors are grateful for the support of Enterprise Ireland under Project CFTD/05/222a. The second author also acknowledges the support of a Marie Curie Transfer of Knowledge Fellowship of the European Community's Sixth Framework Programme, Contract MTKD-CT-2005-029611. Both authors thank Shehroz Khan for his comments and input.

References

1. Bishop C.M.: Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.
2. Tax D.M.J. and Duin R.P.W.: Support vector domain description, Pattern Recognition Letters, 20 (1999), 1191-1199.
3. Manevitz L. M. and Yousef M.: One-class SVMs for Document Classification, Journal of Machine Learning Research, 2(2001), 139-154.

4. Schölkopf B., Williamson R., Smola A., Shawe-Taylor J., Platt J.: Support Vector Method for Novelty Detection. Advances in Neural Information Processing Systems. 12(2000) 582-588.
5. Schölkopf B., Williamson R., Smola A., Shawe-Taylor J., Platt J.: Estimating the Support of a High-dimensional Distribution, Neural Computation, 13 (2001) 1443-1471.
6. Liu Y., Khu S.T.:, Automatic Calibration of Numerical Models Using Fast Optimisation by Fitness Approximation. 2007 International Joint Conference on Neural Networks (IJCNN). (2007).
7. Kennedy J., and Eberhart R.: Particle Swarm Optimisation, in Proc. of the IEEE Int. Conf. on Neural Networks, (1995) 1942–1945.
8. Eberhart R., Shi Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. Washington, DC, (2000) 84-88.
9. Hu X.H.: Particle Swarm Optimisation tutorial, www.swarmintelligence.org/tutorials.php, available online on 26th, Jun. 2007.
10. Hsu C.W, Chang C.C and Lin C, J.: A Practical Guide to Support Vector Classification, Department of Computer Science, National Taiwan University.

Increasing the Coverage of Decision Trees through Mixed-Initiative Interaction

David McSherry

School of Computing and Information Engineering, University of Ulster
Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

Abstract. Inadequate coverage of problems that can be solved by other methods without knowing the result of a test that carries high risk or cost is a serious limitation of decision trees. We present an approach to increasing coverage by a mixed-initiative approach to problem solving in which the user can answer *unknown* to any question in the decision tree or answer questions anywhere in the decision tree without waiting to be asked. However, the potential benefits of allowing problem-solving dialogues to continue when relevant test results are not available must be balanced against the risk that no solution may be possible no matter what additional information the user can provide. This problem is addressed in our approach by using meta-level reasoning to recognize when no solution is possible.

Keywords: intelligent systems, decision trees, coverage, mixed-initiative interaction, meta-level reasoning

1 Introduction

Benefits of mixed-initiative interaction in intelligent systems include involving the user more closely in the problem-solving process and the ability to adapt more easily to the experience level of the user [1-4]. Problem-solving efficiency may also benefit from the experience and knowledge that users can often bring to bear [5]. As we show in this paper, increased coverage of problems for which a complete description is not available is another potential benefit of mixed-initiative interaction.

Fig. 1 shows an example decision tree based on the contact lenses dataset, a simplified version of the real-world problem of selecting a suitable type of contact lenses for an adult spectacle wearer [6]. The attributes are tear production rate (TPR), astigmatism, age, and spectacle prescription. The outcome classes are no contact lenses (N), soft contact lenses (S), and hard contact lenses (H). The traditional approach to problem solving based on such a decision tree is to ask the user a sequence of questions, starting at the root node, and following the path determined by the user's answers until a leaf node is reached. The solution is the outcome class at the leaf node.

However, no solution is possible if the user is unable to answer a question that must be answered to reach a leaf node. This is a serious limitation when the problem can be solved by other methods without knowing the result of a test that carries high risk or cost (e.g., TPR in the contact lenses decision tree). Cendrowska [6] proposes

an approach to addressing this issue in which maximally general (MG) rules are induced from the contact lenses dataset instead of a decision tree. The MG rules often enable a solution to be reached when the TPR, or another test result, is unknown. However, the approach requires access to the original data, which is not an option for a decision tree constructed by a domain expert. It also leaves open the question of how to apply the MG rules in interactive problem solving. Another approach that requires access to the original dataset is dynamic (or *lazy*) induction of decision trees [5, 7]. In this approach, the system delays commitment to the most useful question until it has determined that the user can provide an answer. If not, the dialogue moves on to the next best question.

McSherry [4] proposes a different approach in which a committee of classifiers is applied to the sub-trees for all possible values of an attribute whose value is unknown to the user. The idea is that if all classifiers return the same outcome class, then the attribute whose value is unknown cannot affect the solution. As demonstrated in a mixed-initiative intelligent system called *Confirm*, the approach has the advantage that access to any data used to construct a decision tree is not required. However, an important issue not addressed in this and other approaches is the risk that if a relevant test result is not available, then no solution may be possible no matter what additional information the user can provide. Thus allowing a problem-solving dialogue to continue when relevant tests are not available may result in frustration for the user as well as needlessly incurring the risks and costs of additional tests.

To address these issues, we present a mixed-initiative approach to problem solving based on decision trees in which meta-level reasoning [8-10] is used to ensure that a problem-solving dialogue is allowed to continue only if a solution may still be possible. Our approach also differs from previous work [4] in that the solution of problems for which a complete description is not available is guided by rules in the decision tree rather than exploration of sub-trees. In Sections 2 and 3, we present the theory on which our approach is based. In Section 4, we demonstrate our approach in a mixed-initiative intelligent system called *Confirm*-2. In Section 5, we investigate the coverage benefits of mixed-initiative interaction in the contact lenses domain, and our conclusions are presented in Section 6.

2 Confirming an Outcome Class

Often in mixed-initiative intelligent systems, a query (or problem description) is incrementally elicited (or extended) with the aim of minimizing the number of questions (or tests) required to reach a solution [2-5, 7, 11-13]. The approach we present in this paper assumes the existence of a decision tree D that the system uses to guide its selection of questions when given the initiative by the user. In the traditional approach to problem solving based on decision trees, the dialogue continues until a leaf node is reached. However, this simple stopping criterion does not apply in a mixed-initiative dialogue in which the user can answer *unknown* to any question and/or volunteer information without waiting to be asked. In this section, we describe the criteria used in our approach to recognize when an incomplete problem description provides sufficient evidence for the dialogue to be successfully terminated.

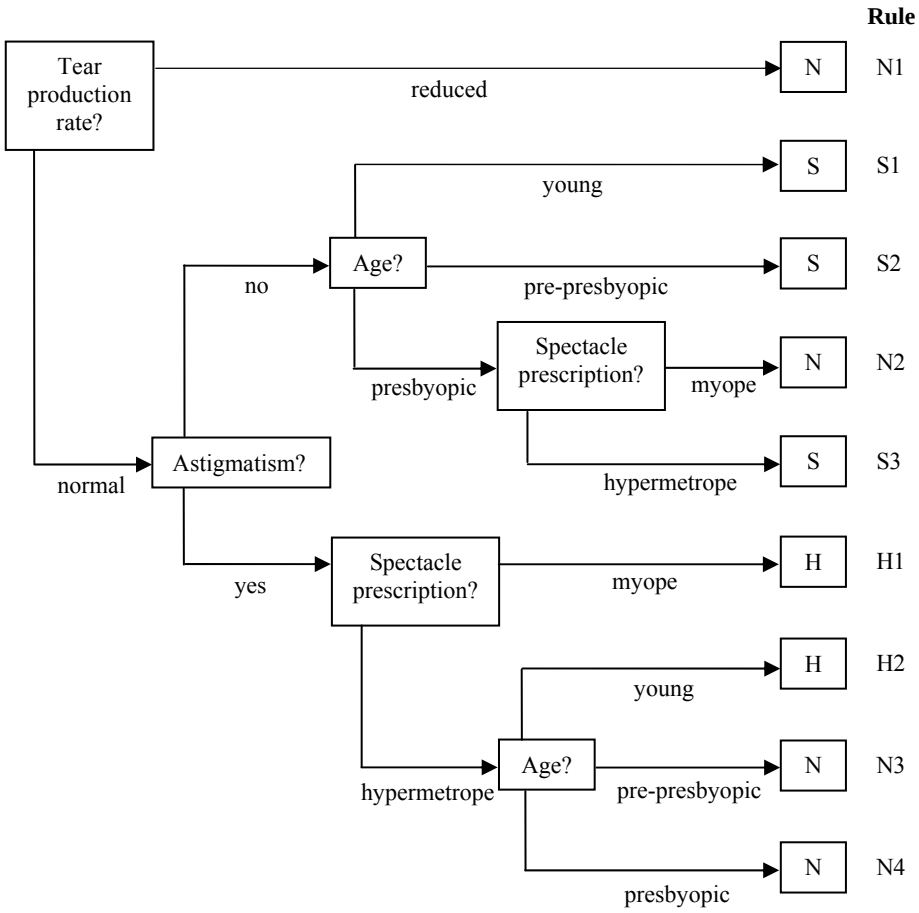


Fig. 1. A decision tree based on the contact lenses dataset

Definition 1. We denote by A_D the set of attributes in the decision tree D.

Definition 2. A query is a set of attribute-value pairs $Q = \{a_1 = v_1, a_2 = v_2, ..., a_n = v_n\}$ such that $a_i \in A_D$ and $v_i \in \mathit{domain}(a_i) \cup \{\text{unknown}\}$ for $1 \leq i \leq n$.

Definition 3. We denote by A_Q the set of attributes, if any, in a given query Q and for each $a \in A_Q$, we denote by $\pi_a(Q)$ the value of a in Q.

Each path from the root node of a decision tree to a leaf node provides a rule for the outcome class at the leaf node. For example, one of the nine rules (N1-N4, S1-S3, and H1-H2) in the contact lenses decision tree (Fig. 1) is:

S1: **if** TPR = normal **and** astigmatism = no **and** age = young **then** S

Definition 4. We denote by *rules*(D) the set of rules in D.

Definition 5. For each $R \in$ *rules*(D), we denote by A_R the attributes in the rule, by *conditions*(R) its conditions, and by *conclusion*(R) its conclusion.

Definition 6. For each $R \in$ *rules*(D), and $a \in A_R$, we denote by $\pi_a(R)$ the required value of a in R.

Some of the questions in a decision tree may be meaningful only in the context of the user's answers to previous questions. However, we assume in this paper that any combination of values of attributes in the decision tree is possible. We also assume that the question at a non-leaf node asks the user for the value of an attribute with a limited number of values, and each of the attribute's values leads to another question node, or to a leaf node at which a solution to the problem is provided. Such a decision tree provides full coverage of the problem space defined by the product of its attribute domains. That is, for any problem $P \in \prod_{a \in A_D} domain(a)$, there exists a unique rule $R_P \in$ *rules*(D) such that *conditions*(R_P) $\subseteq P$. We will refer to *conclusion*(R_P) as the "standard" decision-tree solution for P.

However, if an attribute value required to reach a leaf node of the decision tree is unknown to the user, there can be no rule $R \in$ *rules*(D) such that *conditions*(R) $\subseteq P$, where P is the partial description of the problem that is known to the user. For example, the problem represented by the following query cannot be solved by the traditional approach using the decision tree in Fig. 1:

$Q°$ = {TPR = normal, astigmatism = yes, spectacle prescription = unknown, age = young}

Our approach to increasing coverage of problems for which a complete description is not available is based on the idea that a solution may be possible by using more than one of the rules in *rules*(D). Our criterion for a successful solution is that the incomplete problem description provides sufficient evidence to *confirm* one of the outcome classes in the decision tree. For an outcome class to be confirmed, it must be the standard decision-tree solution for all *complete* problem descriptions that include all the known facts. That is, any attributes that do not have known values cannot affect the solution.

Definition 7. A query Q^* is a *completion* of a given query Q if $A_{Q^*} = A_D$, $\pi_a(Q^*) \in domain(a)$ for all $a \in A_{Q^*}$, and $\pi_a(Q^*) = \pi_a(Q)$ for all $a \in A_Q$ such that $\pi_a(Q) \neq$ unknown.

Definition 8. An outcome class G is confirmed by a given query Q if G is the standard decision-tree solution for all completions Q^* of Q.

In Theorem 1, we present a necessary and sufficient condition for an outcome class to be confirmed by a given query which does not require exhaustive search over all possible completions of the query to determine if a solution has been reached. Instead, it is necessary only to identify "matching" rules for the given query and their conclusions. The proofs of Theorems 1-5 are omitted because of limited space.

Definition 9. For any query Q, we define *matching-rules*(Q) = $\{R \in$ *rules*(D) : $\pi_a(R) = \pi_a(Q)$ for all $a \in A_R \cap A_Q$ such that $\pi_a(Q) \neq$ *unknown*$\}$.

Definition 10. For any query Q, we define *competitors*(Q) = $\{$*conclusion*(R) : $R \in$ *matching-rules*(Q)$\}$.

Theorem 1. *An outcome class G is confirmed by a given query Q if and only if* $competitors(Q) = \{G\}$.

For the example query $Q°$, it can be seen from Fig. 1 that $matching\text{-}rules(Q°) = \{H1, H2\}$. As $competitors(Q°) = \{H\}$, it follows from Theorem 1 that the evidence provided by $Q°$ is sufficient to confirm H in our approach. Thus a problem that cannot be solved by the traditional decision-tree approach can now be solved by making use of two of the rules available in the decision tree. In Section 5, we empirically investigate the overall increase in coverage provided by our approach when applied to the contact lenses decision tree (Fig. 1).

3 Recognizing Non-Confirmable Outcome Classes

In this section, we present our approach to recognizing when no outcome class can be confirmed no matter what additional information the user can provide. An important benefit is that the user can be informed at the earliest possible stage when a solution cannot be reached, and thus avoid the risks and costs of additional tests. We begin by introducing the concepts on which our approach is based, such as matching rules that are *open* and *closed* with respect to a given query, and the *supporters* and *opposers* of an outcome class among the matching rules for a given query.

Definition 11. A query Q is *inconclusive* if $|competitors(Q)| > 1$.

Definition 12. A query Q^+ is an *extension* of another query Q if $Q \subseteq Q^+$.

Definition 13. For any query Q and $R \in matching\text{-}rules(Q)$, we denote by Q^R the extension of Q such that $A_{Q^R} = A_Q \cup A_R$ and $\pi_a(Q^R) = \pi_a(R)$ for all $a \in A_R - A_Q$.

Definition 14. For any query Q, we define $closed(Q) = \{R \in matching\text{-}rules(Q) : A_R \subseteq A_Q\}$ and $open(Q) = \{R \in matching\text{-}rules(Q) : A_R - A_Q \neq \emptyset\}$.

Definition 15. For any outcome class G and query Q, we define $supporters(G, Q) = \{R \in matching\text{-}rules(Q) : conclusion(R) = G\}$ and $opposers(G, Q) = \{R \in matching\text{-}rules(Q) : conclusion(R) \neq G\}$.

In Theorem 2, we present criteria which sometimes enable non-confirmable outcome classes to be easily identified, but which cannot be guaranteed to identify all such outcome classes.

Theorem 2. *An outcome class G can be confirmed by extending an inconclusive query Q only if it is supported by an open rule and not opposed by any closed rule.*

As an example of how Theorem 2 can be used to recognize when no outcome class can be confirmed, consider the query $Q_1 = \{age = young, TPR = unknown\}$. It can be seen from Fig. 1 that $matching\text{-}rules(Q_1) = \{N1, S1, H1, H2\}$, $competitors(Q_1) = \{N, S, H\}$, $closed(Q_1) = \{N1\}$, $open(Q_1) = \{S1, H1, H2\}$, $supporters(N, Q_1) = \{N1\}$, $opposers(S, Q_1) = \{N1, H1, H2\}$, and $opposers(H, Q_1) = \{N1, S1\}$. By Theorem 2, neither H nor S can be confirmed by extending Q_1 as they are both opposed by the closed rule N1. Also by Theorem 2, N cannot be confirmed as there is no open rule that supports it.

The conditions in Theorem 2 are necessary but not sufficient for an outcome class to be confirmable by extending an inconclusive query. For example, it can be seen from Fig. 1 that for Q_2 = {TPR = normal, astigmatism = unknown}, *matching-rules*(Q_2) = *open*(Q_2) = {S1, S2, N2, S3, H1, H2, N3, N4}, *competitors*(Q_2) = {N, S, H}, and *closed*(Q_2) = ∅. Thus N, S, and H are all supported by open rules and unopposed by any closed rule. As will be seen from Theorem 3, however, no outcome class can be confirmed by extending Q_2.

Theorem 3. *An outcome class G can be confirmed by extending an inconclusive query Q if and only if there exists* $R \in$ *supporters*(G, Q) $\cap$ *open*(Q) *such that* Q^R *confirms G or can be extended to confirm G.*

For example, *supporters*(S, Q_2) $\cap$ *open*(Q_2) = {S1, S2, S3}, and it can be seen from Fig. 1 that $Q_2^{S1} = Q_2 \cup$ {age = young}, $Q_2^{S2} = Q_2 \cup$ {age = pre-presbyopic}, and $Q_2^{S3} = Q_2 \cup$ {age = presbyopic, spectacle prescription = hypermetrope}. Table 1 shows all matching, closed, and open rules for Q_2^{S1}, Q_2^{S2}, and Q_2^{S3}. It is clear that S is not confirmed by Q_2^{S1}, Q_2^{S2}, or Q_2^{S3} as they all have two or more competing outcome classes. It can also be seen from Theorem 2 that none of Q_2^{S1}, Q_2^{S2}, Q_2^{S3} can be extended to confirm S as none of them has an open rule that supports S. It follows from Theorem 3 that no extension of Q_2 can confirm S. It can similarly be verified that neither N nor H can be confirmed by extending Q_2. The user can thus be informed that no solution is possible because astigmatism is unknown.

Table 1. Matching, closed, and open rules for Q_2^{S1}, Q_2^{S2}, and Q_2^{S3}

Extension	*Matching Rules*	*Closed Rules*	*Open Rules*
Q_2^{S1}	{S1, H1, H2}	{S1}	{H1, H2}
Q_2^{S2}	{S2, H1, N3}	{S2}	{H1, N3}
Q_2^{S3}	{S3, N4}	{S3, N4}	None

For an inconclusive query Q and outcome class G that is supported by an open rule R but not confirmed by Q^R, it is immediate from Theorem 2 that Q^R cannot be extended to confirm G if no rule that is open with respect to Q^R supports G, or if G is opposed by a rule that is closed with respect to Q^R. If neither condition holds, it remains to be determined if G can be confirmed by extending Q^R. In Theorem 4, we show how this task can be formulated as a search for a sequence of rules that confirms G. It is worth noting that the search involves only open rules that support G.

Theorem 4. *An outcome class G can be confirmed by extending an inconclusive query Q if and only if there exists* $R_1 \in$ *supporters*(G, Q) $\cap$ *open*(Q) *such that* Q^{R_1} *confirms G or a sequence of rules* $R_1, \ldots, R_k \in$ *supporters*(G, Q) $\cap$ *open*(Q) *such that* $R_i \in$ *supporters*(G, $Q^{R_1 \cdots R_{i-1}}$) $\cap$ *open*($Q^{R_1 \cdots R_{i-1}}$) *for* $i \geq 2$ *and* $Q^{R_1 \cdots R_k}$ *confirms G.*

In the final theorem of this section, we show that any competing outcome class (Section 2) can be confirmed for a query in which all attributes have *known* values.

Theorem 5. *If all attributes in an inconclusive query Q have known values, then an outcome class G can be confirmed by extending Q if and only if* $G \in$ *competitors*(Q).

4 Mixed-Initiative Interaction in Confirm-2

We now use the contact lenses decision tree (Fig. 1) to demonstrate our approach in *Confirm*-2, a mixed-initiative intelligent system in which the user can provide an initial query, answer *unknown* to any question, and volunteer additional information at any stage without waiting to be asked. Questions are selected by *Confirm*-2 with the goal of confirming a *target* outcome class. In the problem-solving cycle shown in Fig. 2, the identification of confirmable outcome classes (Step 2) is guided by *meta-rules* based on the theoretical results presented in Section 3:

Meta-Rule 1: **if** all attributes in Q have known values and $G \in competitors(Q)$ **then** G can be confirmed

Meta-Rule 2: **if** G is opposed by a closed rule **then** G cannot be confirmed

Meta-Rule 3: **if** G is not supported by an open rule **then** G cannot be confirmed

Meta-Rule 4: **if** no sequence of open rules confirms G **then** G cannot be confirmed

If there is at least one confirmable outcome class, then the one supported by most rules that match the current query is selected as a target outcome class (Steps 4-5). The open rule with fewest conditions among those that support the target outcome

---

Algorithm: Mixed-Initiative(Q)

Repeat Steps 1-10 **until** one of the stopping criteria in Steps 1 and 3 is satisfied:

1. If $competitors(Q) = \{G\}$, where G is a single outcome class, then inform the user that G has been confirmed and stop
2. Identify all confirmable outcome classes in $competitors(Q)$:
   $$targets(Q) \leftarrow \{G \in competitors(Q) : Q \text{ can be extended to confirm } G\}$$
3. If $targets(Q) = \varnothing$ then inform the user that no outcome class can be confirmed and stop
4. Identify the outcome classes in $targets(Q)$ that are supported by most rules in $matching\text{-}rules(Q)$:
   $$best\text{-}targets(Q) \leftarrow \{G^* \in targets(Q) : |supporters(G^*, Q)| \geq |supporters(G, Q)| \text{ for all } G \in targets(Q)\}$$
5. Select any $G^* \in best\text{-}targets(Q)$ as the target outcome class
6. Identify the open rules in $supporters(G^*, Q)$ that have fewest conditions:
   $$best\text{-}rules(G^*) \leftarrow \{R^* \in supporters(G^*, Q) \cap open(Q) : |conditions(R^*)| \leq |conditions(R)| \text{ for all } R \in supporters(G^*, Q) \cap open(Q)\}$$
7. Select any $R^* \in best\text{-}rules(G^*)$ as the most useful rule
8. Select the first attribute a^* in R^* such that $a^* \notin A_Q$ as the most useful attribute
9. Ask the user for the value of a^*
10. If v^* is the value of a^* reported by the user then $Q \leftarrow Q \cup \{a^* = v^*\}$ *else* if the value of a^* is unknown to the user then $Q \leftarrow Q \cup \{a^* = \text{unknown}\}$ *else* if v is the value of another attribute a reported by the user then $Q \leftarrow Q \cup \{a = v\}$

Fig. 2. The problem-solving cycle in *Confirm*-2

class is selected as the most useful rule (Steps 6-7). The first attribute in this rule that is not already in the current query is selected as the most useful attribute (Step 8). When asked for the value of the selected attribute, the user can answer *unknown* or select another attribute whose value she wishes to report (Steps 9-10).

Fig. 3 shows a *Confirm*-2 dialogue in which the initial query is Q_1 = {age = presbyopic}. It can be seen from Fig. 1 that *matching-rules*(Q_1) = *open*(Q_1) = {N1, N2, S3, H1, N4}, so *competitors*(Q_1) = {N, S, H}. As all attributes in Q_1 have known values, it follows from Meta-Rule 1 that N, S, H can all be confirmed by extending Q_1, and so *targets*(Q_1) = {N, S, H} (Step 2). As N, S, and H are supported by 3, 1, and 1 matching rules respectively, *best-targets*(Q_1) = {N} (Step 4) and the target outcome class is G^* = N (Step 5). As *supporters*(N, Q_1) $\cap$ *open*(Q_1) = {N1, N2, N4} and N1, N2, and N4 have 1, 4, and 4 conditions respectively, *best-rules*(N) = {N1} (Step 6) and the most useful rule is R^* = N1 (Step 7).

As the only attribute in N1 is a^* = TPR, the user is now asked for the TPR (Step 9). As the TPR is unknown to the user (Fig. 3), the problem-solving cycle is repeated with Q_2 = {age = presbyopic, TPR = unknown} as the current query. From Fig. 1, *matching-rules*(Q_2) = {N1, N2, S3, H1, N4}, *closed*(Q_2) = {N1}, and *open*(Q_2) = {N2, S3, H1, N4}. As S and H are now opposed by the closed rule N1, it follows from Meta-Rule 2 that they cannot be confirmed. However, N is confirmable by Q_2^{N2} = {age = presbyopic, TPR = unknown, astigmatism = no, spectacle prescription = myope} as *matching-rules*(Q_2^{N2}) = {N1, N2} and *competitors*(Q_2^{N2}) = {N}. So *targets*(Q_2) = {N}, *best-targets*(Q_2) = {N}, and G^* = N. As *supporters*(N, Q_2) $\cap$ *open*(Q_2) = {N2, N4} and N2 and N4 both have 4 conditions, *best-rules*(N) = {N2, N4}. With R^* = N2 now as the most useful rule, the most useful attribute is a^* = astigmatism.

The user is now asked if astigmatism is present (Fig. 3), and a third cycle begins with Q_3 = {age = presbyopic, TPR = unknown, astigmatism = no} as the current query. In this cycle, *matching-rules*(Q_3) = {N1, N2, S3}, *competitors*(Q_3) = {N, S}, *closed*(Q_3) = {N1}, *open*(Q_3) = {N2, S3}, *targets*(Q_3) = *best-targets*(Q_3) ={N}, G^* = N, *best-rules*(N) = {N2}, R^* = N2, and a^* = spectacle prescription. The user is now asked for the patient's spectacle prescription, and the dialogue enters a fourth cycle with Q_4 = {age = presbyopic, TPR = unknown, astigmatism = no, spectacle prescription = myope} as the current query. As *matching-rules*(Q_4) = {N1, N2} and *competitors*(Q_4) = {N}, the user is informed that N (no contact lenses) has been confirmed and the problem-solving process is successfully terminated (Step 1).

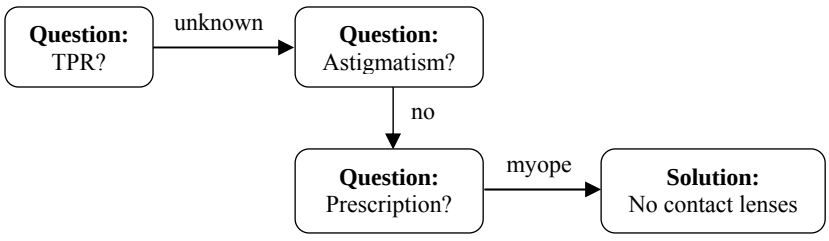


Fig. 3. Example dialogue with Q_1 = {age = presbyopic} as the initial query in *Confirm*-2

5 Empirical Study

Our evaluation focuses on the coverage benefits of mixed-initiative problem solving based on the contact lenses decision tree (Fig. 1) in comparison with the traditional decision-tree approach. A problem for which a complete description is not available can be solved in the traditional approach if the incomplete description known to the user includes all the conditions of one of the rules in the decision tree. It can be solved in our mixed-initiative approach if all rules in the decision tree that match the incomplete problem description have the same conclusion. Initially we compare the percentages of successful solutions over all possible problem descriptions, both complete and incomplete, in the two approaches. Attributes in the domain and their numbers of values are TPR (2), astigmatism (2), age (3), and spectacle prescription (2). The number of problem descriptions in which the values of one or more attributes may be unknown is therefore: $(2 + 1) \times (2 + 1) \times (3 + 1) \times (2 + 1) = 108$.

All such problem descriptions were automatically generated and tested for the existence of a unique solution in the two approaches. As shown by the results for all problem descriptions (n = 108) in Fig. 4, mixed-initiative interaction increases the percentage of successful solutions from 47% to 52%. However, as problems for which a complete description is available (n = 24) can always be solved in both approaches, the benefits of mixed-initiative interaction can be seen more clearly from the results for incomplete problem descriptions. For incomplete problem descriptions (n = 84), mixed-initiative interaction increases the percentage of successful solutions from 32% to 38%, which amounts to a relative increase in coverage of 19%.

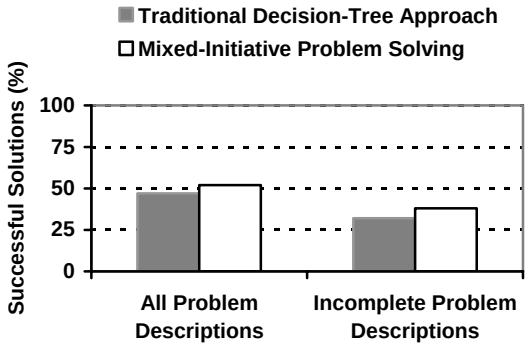


Fig. 4. Problems in the contact lenses domain that can be solved by the traditional decision-tree approach and by mixed-initiative problem solving

6 Conclusions

We have presented a mixed-initiative approach to increasing decision-tree coverage of problems for which a complete description is not available. In the classification task that we studied, mixed-initiative interaction was shown to increase coverage of problems for which the user is unable to provide a complete description by 19%.

However, a potential risk in allowing problem-solving dialogues to continue when relevant test results are not available is that no solution may be possible no matter what additional information the user can provide. Our solution to this problem is based on the use of meta-level reasoning to identify outcome classes that cannot be confirmed by extending the user's current query. As shown by our theoretical results, this enables the user to be informed at the earliest possible stage when a solution cannot be reached, thus avoiding the risks and costs of additional tests.

In future work, we will investigate the coverage benefits of mixed-initiative interaction when applied to more complex decision trees. Of course, our approach can be expected to provide little or no increase in coverage if most or all of the outcome classes at the leaf nodes of the decision tree are distinct. However, the ability to recognize when no solution is possible (Section 3) remains an important benefit of our approach even when the achievable coverage gains are small.

References

1. Allen, J.E.: Mixed-Initiative Interaction. IEEE Intelligent Systems **6** (1999) 14-16
2. Cheetham, W.: A Mixed-Initiative Call Center Application for Appliance Diagnostics. In: Aha, D.W., Tecuci, G. (eds.): Proceedings of the AAAI-05 Fall Symposium on Mixed-Initiative Problem-Solving Assistants. AAAI/MIT Press (2005)
3. Göker, M.H.: Adapting to the Level of Experience of the User in Mixed-Initiative Web Self-Service Applications. ICCBR-03 Workshop on Mixed-Initiative Case-Based Reasoning (2003)
4. McSherry, D.: Mixed-Initiative Intelligent Systems for Classification and Diagnosis. Proceedings of the 14th Irish Conference on Artificial Intelligence and Cognitive Science (2003) 146-151
5. McSherry, D.: Interactive Case-Based Reasoning in Sequential Diagnosis. Applied Intelligence **14** (2001) 65-76
6. Cendrowska, J.: PRISM: an Algorithm for Inducing Modular Rules. International Journal of Man-Machine Studies **27** (1987) 349-370
7. Bergmann, R., Cunningham, P.: Acquiring Customers' Requirements in Electronic Commerce. Artificial Intelligence Review **18** (2002) 63-193
8. Cox, M.: Metacognition in Computation: A Selected Research Review. Artificial Intelligence **169** (2005) 104-141
9. Lenat, D., Davis, R., Doyle, J., Genesereth, M., Goldstein, I., Schrobe, H.: Reasoning about Reasoning. In: Hayes-Roth, F., Waterman, D. Lenat, D. (eds.): Building Expert Systems. Addison-Wesley, Reading, Massachusetts (1983)
10. McLaren, B., Ashley, K.: Helping a CBR Program to Know What it Knows. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer, Berlin (2001) 377-391
11. Aha, D.W., McSherry, D., Yang, Q.: Advances in Conversational Case-Based Reasoning. Knowledge Engineering Review **20** (2005) 247-254
12. Shimazu, H., Shibata, A., Nihei, K.: ExpertGuide: a Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces. Applied Intelligence **14** (2001) 33-48
13. Thompson, C.A., Göker, M.H., Langley, P.: A Personalized System for Conversational Recommendations. Journal of Artificial Intelligence Research **21** (2004) 393-428

Evolving a Hybrid Deceptive Strategy for the Repeated English Auction

Pilib Ó Broin and Colm O'Riordan

Dept. of Information Technology,
National University of Ireland, Galway,
pilib@zig.it.nuigalway.ie, colm.oriordan@nuigalway.ie

Abstract. Understanding issues of trust and deception are key to designing robust, reliable multi-agent systems. This paper builds on previous work which examined the use of auctions as a model for exploring the concept of deception in such systems. We have previously described two forms of deceptive behaviour which can occur in a simulated repeated English auction. In this work we examine the potential shortcomings of those two strategies and investigate whether or not their individual strengths can be combined to produce a successful hybrid deceptive strategy.

1 Introduction

Deception can play a role in any form of interaction, especially when the interaction involves two or more self-interested parties. Recent research has shown that in online interactions the temptation to deceive is particulary strong, as users feel protected by anonymity [1]. In open multi-agent systems, where autonomous agents are often required to try and maximize their own utility, there are many motivations to deceive [2].

Auctions have previously been used to examine the behaviour of rational trading agents [3] and represent an interesting real-world model which has become increasingly popular in recent years due to the success of websites such as Ebay and Amazon. Deception in real-world auctions is surprisingly common and interested readers are directed to [4] which describes some of the many different forms of deception which can occur. Deception is also an important research topic in multi-agent systems, examples of which can be seen in [2][5].

Previously we have shown how auctions can be used as a simple mechanism for investigating notions of deception in a multi-agent system [6]. We have examined two forms of deceptive behaviour which can evolve in a repeated English auction setting — sniping and antisocial bidding.

In this paper we investigate whether or not a combination of these two strategies can produce a hybrid strategy which will perform better than either (or both) of its predecessors. As before, we will examine both the placement of the individual strategy types in the population fitness rankings as well as the overall societal fitness over time.

The next section briefly describes the two strategies previously implemented and outlines their respective limitations. Section 3 deals with the experimental setup including game design and strategy implementation, while Section 4 presents the results of the experiments and their analysis. The final section includes a summary and general discussion and outlines future work.

2 Deceptive Strategies

2.1 Sniping and Antisocial Bidding

Sniping, or late bidding, provides a means for an agent to hide information about its true valuation for an item, while simultaneously allowing it to win items at a price below that which it would normally be forced to pay had a 'Naïve' (honest) strategy been employed. Much recent research has focussed on sniping which is widespread in real-world online auctions such as ebay [7][8].

Antisocial bidding was first introduced in [9]. Based on the notion of relative fitness (concentrating not only on maximzing one's own fitness, but also on minimizing that of other agents), strategies employing antisocial bidding place 'false bids' (bids for items which they do not wish to acquire). These bids, when placed on an item, serve to ensure that, over time, an opponent will be forced to pay an amount increasingly close to their private valuation in order to win that item. This has the net effect of reducing the payoff for the agent who wins that item[1] and thus reducing their chance of survival. Previous results showed that antisocial bidders, (herefter refered to as *Dec2*), effectively reduce the payoffs of competitors to zero on many items [6]. A form of antisocial bidding has also been documented in the real world where sellers in online auctions use multiple accounts to uncover buyers' private valuations for items and then place false or 'shill' bids just below those private valuations in order to increase their own profit. The authors in [4] term this behaviour 'squeezing'.

Both the sniping and *Dec2* strategies were shown to achieve higher payoffs than the test population of Naïve strategies into which they were introduced. These high levels of payoff ensured that the deceptive strategies were selected for reproduction and quickly spread among the Naïve agents. In co-evolutionary settings, snipers were seen to be somewhat robust to the profit-reducing effect of the *Dec2* strategy and populations usually converged to the former within 15 generations. This result coincides with the findings in [4], which indicate that sniping is a possible best response to the real-world scenario of 'squeezing'.

2.2 Motivation for Hybrid Strategy

Although both types of strategy successfully exploit Naïve populations, they each implement only one form of deception. Snipers, while winning items for which they do not possess the highest private valuation, do nothing to reduce

[1] Payoffs are calculated as the difference between an agent's private valuation for an item and the price actually paid to win that item.

the profits of their opponents. This is a missed opportunity for increasing their relative fitness.

Dec2 bidders, on the other hand, focus solely on decreasing their opponents' profit, they do not try to use late bidding in order to win items they wish to acquire at lower prices. A successful hybrid strategy would need to incorporate both types of deception in order to fully exploit the weaknesses of its opponents.

3 Experimental Setup

This section examines the game model including the auction mechanism used and the strategy set design.

3.1 Game Design

As before, agents compete in a repeated English auction (incremental bidding) with hard close (fixed end time). Each agent is assigned the same budget and a random subset of the complete set of auction items, this is refered to as the agent's *goal list*. Each item on this list is randomly assigned a portion of the agent's budget which corresponds to the agent's *private valuation* for that item. Each auction has ten timeslots in which bids can be placed on any item. At the end of an auction agents are assigned fitness scores based on their level of profit relative to the profit of their opponents.

A single game in the simulator consists of seven auctions, and the agents partake in 500 games per generation. Different goal lists are assigned to the agents in each game, ensuring that strategies must be able to compete effectively accross a wide variety of auction scenarios.

3.2 Strategy Design

The bidding behaviour of Naïve agents is encoded by so-called *naïve genes*. These three genes — *rand*, *linear* and *dyn* control the agressiveness of bidding in the Naïve strategy.

The behaviour of a strategy defined as being deceptive is based on three additional *deceptive genes*. The first deceptive gene – *dec*, represents the type of deception used by the agent. In experiments with a single type of deceptive agent, a value for this gene of less than 0.5 indicates a Naïve strategy, while anything above this value indicates a deceptive one. When more than one type of deceptive strategy is involved in an experiment, the dec gene range ([0..1]) is split three ways, allowing us to encode each of the three types of deceptive strategy.

The remaining two deceptive genes code for the behaviour of the Sniper (or Hybrid strategy which incorporates sniping). These genes – *bid-time* and *bid-amount*, control how late in the auction bids are placed (i.e. a bid-time of 0.7 indicates bidding in the seventh timeslot), as well as the value of those bids. The *Dec2* strategy ignores the bid-time and bid-amount genes and instead uses its

naïve genes for bidding on items on its goal list. Its additional data structures allow it to maintain a limited bid history which it uses to calculate the false bids that it will place.

4 Results and Analysis

The Hybrid strategy is tested against our three previous strategies (Naïve, Sniper and *Dec2*). We begin by testing the performance of the Hybrid strategy against Naïve bidders in both non-evolutionary and evolutionary settings. We then show the results following a single generation of introducing one Hybrid strategy into populations of Snipers and *Dec2*'s. Finally, we examine the performance of the Hybrid strategy in a population consisting of an equal mix of all three deceptive strategies which is allowed to evolve over 500 generations.

4.1 Hybrid vs. Naïve

Non-evolutionary Setting We begin by testing our Hybrid strategy against Naïve bidders in a non-evolutionary environment to ascertain its position in the fitness rankings after one generation. The test population consits of nine randomly initialized Naïve agents and one Hybrid agent playing 500 games with a goal list of five items each from a complete auction set of ten items. The Hybrid agent has a deceptive gene value of 0.75, while the Naïve strategies' deceptive genes are set to zero. The Hybrid strategy has a bid-time of 1 (indicating that it will bid in the last possible timeslot), and a bid-amount of 0.1.

As we can see in Table 1, the Hybrid strategy (bold) receives a fitness score 0.42 while the next fittest strategy receives a score of 0.17. This fitness is higher than a Sniper (.21) and comparable to a *Dec2* agent (.43) in a similar scenario.

Table 1. Hybrid Position in Naïve Fitness Rankings

Rand	Linear	Dyn	Bt	Ba	Dec	Fitness
0.47	0.95	0.05	0.55	0.42	0	0.0183
0.25	0.80	0.51	0.27	0.29	0	0.0310
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0.39	0.26	0.85	0.03	0.76	0	0.0742
0.07	0.03	0.29	0.82	0.32	0	0.1749
0.14	0.58	0.44	**1**	**0.1**	**0.75**	**0.4280**

In a population of fit (pre-evolved) Naïve strategies, the Hybrid's ability to reduce the profit of its opponents is limited due to the Naïve strategies' use of small bid increments which cause them to reach their true valuations more slowly. The Hybrid strategy's use of sniping, however, ensures that it still consistently achieves the highest fitness score. This is a marked improvement on the *Dec2* strategy's performance demostrated in [6].

Evolutionary Setting Next, we examine the representation of the different strategies in the population as well as the societal fitness over 500 generations for the test population used above.

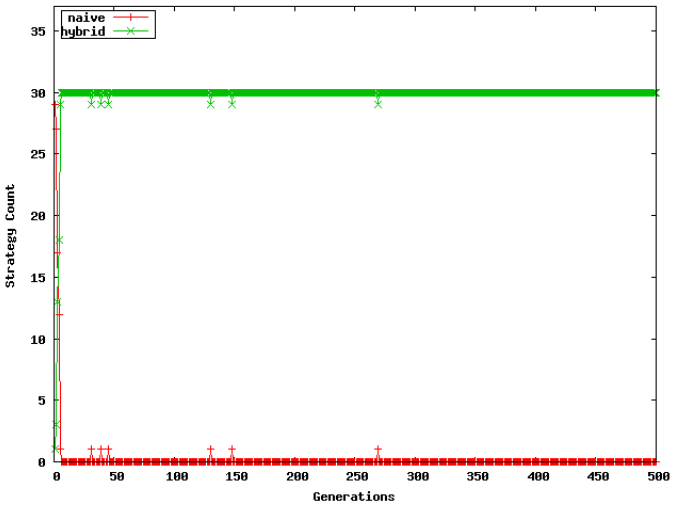


Fig. 1. Strategy Count with Naïve and Hybrid Strategies

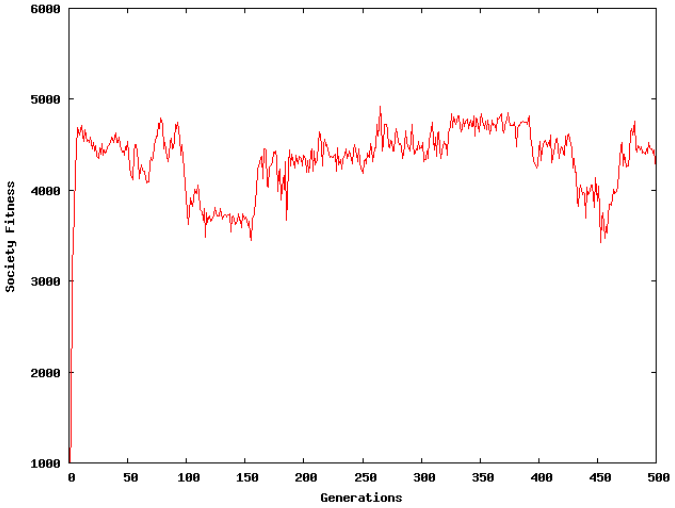


Fig. 2. Societal Fitness with Naïve and Hybrid Strategies

Figure 1 shows the number of Naïve and Hybrid strategies in the population at each generation. The Hybrid strategy quickly spreads and dominates the population for the remainder of the run despite a few Naïve strategies reappearing through mutation.

The effect of this spread of the Hybrid strategy on societal fitness is seen in Figure 2. The sharp rise in the first ten generations is a result of more and more of the agents in the population adopting a Hybrid approach to bidding. As previously seen in the case of Snipers, late bidding means a decrease in the amount paid to win an item and thus a corresponding increase in profit and overall societal fitness. The remainder of the run sees the societal fitness fluctuate as the bid-amount gene undergoes a process of genetic drift as outlined in [6].

4.2 Hybrid vs. Sniper

Table 2. Hybrid Position in Sniper Fitness Rankings

Rand	Linear	Dyn	Bt	Ba	Dec	Fitness
0.48	0.69	0.11	1	0.1	0.17	0.0920
0.24	0.73	0.79	1	0.1	0.17	0.0967
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0.12	0.64	0.58	1	0.1	0.17	0.1024
0.75	0.62	0.17	1	0.1	0.17	0.1057
0.51	0.35	0.66	**1**	**0.1**	**0.83**	**0.1094**

Table 2 shows the fitness rankings after a single generation (500 games) when one Hybrid strategy is introduced into a population of nine Snipers. In this experiment the deceptive gene was used to encode the three different deceptive strategies (Snipers >0, *Dec2* >.33, Hybrid >.66). As such, the strategies were assigned a deceptive gene value roughly halfway through the range of values encoding their individual behaviour (i.e. 0.17 for Snipers and 0.83 for Hybrids). Both the Snipers and Hybrids had a bid-time of 1 and a bid-amount of 0.1.

The Hybrid strategy achieves a slightly higher fitness score (.109) than the fittest Sniper (.105). The sniping of the Hybrid strategy puts it on a par with the Snipers in terms of winning items for which it has a private valuation greater than zero, while the *Dec2* behaviour of placing false bids reduces its opponents' profit. The net result is that the Hybrid strategy wins items at a rate similar to Snipers, but reduces the Snipers' profit while its own remains unaffected thus achieving a higher relative fitness.

4.3 Hybrid vs. *Dec2*

In these experiments, *Dec2* strategies were assigned a deceptive gene value of 0.5, while Hybrid strategies were again encoded by a value of 0.83. The Hybrid

Table 3. Hybrid Position in *Dec2* Fitness Rankings

Rand	Linear	Dyn	Bt	Ba	Dec	Fitness
0	0.02	0.02	0.65	0.05	0.5	0.0871
0	0.02	0.02	0.24	0.60	0.5	0.0888
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0	0.02	0.02	0.90	0.17	0.5	0.0979
0	0.02	0.02	0.09	0.30	0.5	0.0991
0.52	0.28	0.67	**1**	**0.1**	**0.83**	**0.1536**

strategy had a bid-time of 1 and a bid-amount of 0.1. The *Dec2* startegy had its naïve genes set to the final values which were observed in Naïve strategies which were evolved for 500 generations.

As we would expect the *Dec2* strategies have the usual effect on one another (and on the Hybrid strategy) of decreasing an opponent's profit. The Hybrid strategy, however, achieves a higher fitness score due to its use of sniping. As previously shown, the *Dec2* strategy is not as effective in reducing the profit of Snipers as it is in the case of Naïve bidders. This means that the sniping behaviour of the Hybrid strategy gives it an advantage over the *Dec2* strategies in this scenario.

4.4 All Three Deceptive Strategies

This section provides the results of an experiment involving a population of 30 agents consisting of 10 *Dec2*, 10 Sniper and 10 Hybrid strategies.

In the fitness rankings after one generation, the *Dec2* strategy type (deceptive gene value of 0.5) performs the worst, with the ten seeded strategies achieving the ten lowest scores. The Sniper (deceptive gene value of 0.17) and Hybrid (deceptive gene value of 0.83) strategies are randomly ordered in the top 20 positions. This may seem odd considering that the Hybrid strategy achieved the highest score when seeded in a population of snipers, but can be explained by the presence of further Hybrid (as well as *Dec2*) strategies.

When our Hybrid strategy beat each of its sniping competitors in the non-evolutionary setting, it was the only strategy which was placing false bids to decrease it opponents' profit. Since there is more than one strategy incorporating this kind of bidding in this new population, the Hybrid strategies also have their profit reduced. This, in effect, negates the benefit of the *Dec2* component of the Hybrid strategy leaving its success dependent solely on its sniping genes. In essence, this means that a Hybrid strategy in a population with fellow Hybrids, or indeed with *Dec2*'s will perform as though it were simply a Sniper.

Since Snipers with the same bid-time place bids in a random order[2], their success in winning items and thus their fitness is also random. This accounts for

[2] The equivalent of 30 people placing a bid in an online auction at the exact same time

Table 4. All Three Deceptive Strategies – Fitness Rankings

Rand	Linear	Dyn	Bt	Ba	Dec	Fitness
0	0.02	0.02	0.20	0.78	0.5	0.0124
0	0.02	0.02	0.56	0.22	0.5	0.0135
0	0.02	0.02	0.12	0.56	0.5	0.0145
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0.56	0.95	0.46	1	0.1	0.83	0.0430
0.12	0.92	0.32	1	0.1	0.17	0.0432
0.68	0.24	0.31	1	0.1	0.17	0.0439
0.42	0.78	0.22	1	0.1	0.17	0.0440
0.89	0.85	0.19	1	0.1	0.83	0.0445
0.21	0.01	0.37	1	0.1	0.17	0.0463
0.12	0.72	0.45	1	0.1	0.83	0.0468

the mix of Sniper and Hybrid strategies in the top 20 positions in the fitness rankings. The effect that this has on the representation of the various strategies in the population over time is shown in Figure 3.

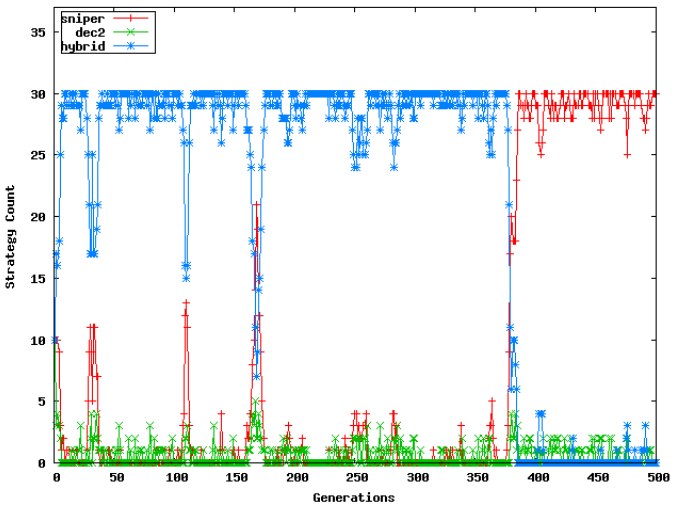


Fig. 3. Strategy Count with All Three Deceptive Strategies

Since the *Dec2*'s consistently achieve the lowest fitness scores, they die off within the first ten generations and do not gain a foothold for the remainder of the run. As the fitness values of the Sniper and Hybrid agents are essentially the same, the population does not fully converge to either of these strategies, but instead fluctuates between the two. In the example shown, the Hybrid strategy

places highest in the fitness rankings after the first generation and so dominates intially. If mutation introduces enough Snipers, or if several Snipers are highly placed in the fitness rankings, then a 'critical mass' is reached in which the evolutionary pressure switches from Hybrids to Snipers (as happens at approximately generation 375). This cycle will continue as long as the population is allowed to evolve and ultimatley neither strategy will remain indefinitely stable.

The societal fitness for this population intially increases as the *Dec2* strategies change to either the Sniper or Hybrid strategies within the first 10-15 generations. After that point the average fitness fluctuates similarly to that of a population consisting entirely of Snipers, although the inclusion of some *Dec2*'s sees a slight reduction in the average fitness due to their decreasing of opponents' profits.

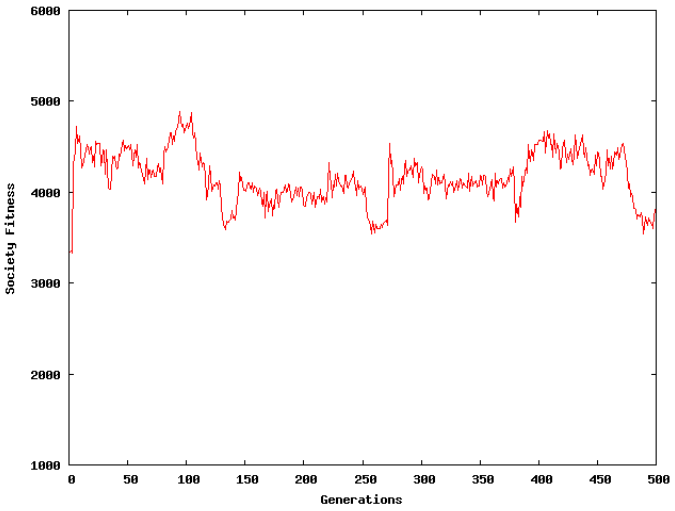


Fig. 4. Societal Fitness with All Three Deceptive Strategies

5 Summary and Conclusions

Understanding issues relating to trust and deception is vital not only in the field of multi-agent systems, but also in areas such as sociology, political science and business, or indeed any other field which involves predicting or reasoning about the behaviour of self-interested parties.

The examining of these concepts can lead, not only to better understanding of their function in a social context, but also, in technological terms, to the design of more reliable systems which would be robust to misuse by malicious behaviour. Any multi-agent system which is designed to constrain deceptive behaviour is inherently more likely to elicit a higher level of trust and confidence in its users.

This paper has examined the limitations of previously discussed deceptive strategies for the repeated English auction. We have proposed a Hybrid deceptive strategy incorporating successful elements from both of these strategies and tested its performance against both Naïve strategies and previous deceptive strategies in an evolutionary setting. The Hybrid strategy was found to have advantages over the Sniper and *Dec2* strategies individually over a single generation, but suffered from the same problem as *Dec2* in an evolutionary setting i.e. when faced with an opponent using the same profit-reducing technique, its performance was diminished. This being said, the Hybrid strategy proved more successful than the *Dec2* strategy and equally as successful as the Sniper in evolutionary settings, with populations converging to the Hybrid strategy in 50% of experiments and to the Sniper strategy in the remaining 50%. The experiment can also be seen as a confirmation of the robust, effectiveness of the Sniper strategy, which can be seen as having the advantages of less complex behaviour and fewer resource requirements.

Future work will focus on examining types of deception possible in alternative auction protocols as well as exploring the possibility of allowing *Dec2* / Hybrid strategies to recognize one another and thus implement bidder collusion in an attempt to address their shared weakness.

References

1. Cristiano Castelfranchi. The role of trust and deception in virtual societies. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.
2. S. D. Ramchurn, T. D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 2004.
3. Michael P. Wellman and Peter R. Wurman. A trading agent competition for the research community. In *Proceedings of the International Joint Conference on Artificial Intelligence, Workshop on Agent-Mediated Electronic Trading*. AAAI Press, 1999.
4. Salvatore Barbaro and Bernd. Bracht. Shilling, squeezing, sniping: Explaining late bidding in online second-price auctions. University of Mainz, working paper (this version: January 31, 2006).
5. C. Castelfranchi, R. Falcone, and F. de Rosis. Deceiving in golem: How to strategically pilfer help. In *Deception, Fraud and Trust in Multiagent Systems*. Kluwer Publishing, 1998.
6. Pilib Ó Broin and Colm O'Riordan. Decption in multi-agent auctions: An evolutionary approach. In *17th Irish Artificial Intelligence and Cognitive Science Conference (AICS2006)*, 2006.
7. A. E. Roth and A. Ockenfels. Last-minute bidding and the rules for ending second-price auctions: Evidence from ebay and amazon auctions on the internet. *American Economic Review*, 92:1093–1103, 2002.
8. A. E. Roth and A. Ockenfels. Late bidding in second price internet auctions: Theory and evidence concerning different rules for ending an auction. *Games andEconomic Behavior*, 50, 2005.
9. F. Brandt. Antisocial bidding in repeated Vickrey auctions. Technical Report FKI-241-00, Department of Computer Science, Technical University of Munich, 2000. ISSN 0941-6358.

Analogy and Sense Extension

Dervla O'Keeffe and Fintan Costello

School of Computer Science and Informatics,
University College Dublin *
dervla.okeeffe@ucd.ie, fintan.costello@ucd.ie

Abstract. We describe an experiment designed to investigate the relationship between sense extension and analogy. Previous accounts of sense extension have been based on the generative lexicon, but it is difficult to see how those accounts explain the everyday apparently analogical sense extensions we see around us. The results of the experiment suggest that the analogical process plays an important role in a type of sense extension, identified by us as analogical sense extension. A positive correlation was found between mappings constructed for analogies and the occurence of sense extension. We did not find evidence of a priming effect when participants were asked to perform analogical processing on the experimental materials prior to answering questions about sense extension.

1 Introduction

Sense extension is the process of extending the meaning of a word to a new domain or area of discourse, so that the word takes on a new meaning. Sense extension is a familiar and integral part of language growth, and results in words having multiple meanings or senses. In this paper we investigate the idea that some forms of sense extension involve the construction and use of analogies.

As an informal example, consider the words associated with aircraft. It may be noticed that many of these words are also used for seacraft. The galley in both an airplane and a ship is the place where food is prepared. A steward on an airplane looks after the food requirements of the passengers, as would a steward on a ship. In both a plane and a ship the cargo is stored in an area known as the hold. An airplane docks at a pier in an airport, and a ship docks at a pier in a port. It seems apparent that the meaning of many of these words has been extended from the domain of seacraft to the domain of aircraft. This sense extension may be enabled because of the analogy between aircraft and seacraft, allowing the words originally used for ships to be used in relation to airplanes.

Current explanations of sense extension, however, are not based on the existence of an analogy between domains, but instead are based on the application of

* This research was supported by the FP6 NEST Programme of the European Commission (ANALOGY: Humans the Analogy-Making Species: STREP Contr. No. 029088)

specific lexical rules. It is, however, difficult to see how they can be applied satisfactorily to the apparently analogical sense extension as described above. Our particular aim in this paper, therefore, is to examine the relationship between analogy and sense extension in the context of these lexical rules.

The structure of the paper is as follows. In Section 2 we provide the necessary background information concerning sense extension and analogy. The experiment is described in Section 3. This experiment investigates the role of analogy in the sense extension of nouns from one domain to another. The results of this experiment and a discussion of the implications of these results are presented in Section 4.

2 Background

2.1 Sense Extension

Polysemy The term *sense extension* is very broad, and is used to describe various different ways in which words can take on new meanings. One type of sense extension is called *regular polysemy*[1, 2], and involves particular standard forms of sense extension such as metonyom and synecdoche. *Metonymy* is the use of a single characteristic to identify a more complex entity. For example, the use of the word "dish" to refer to a meal, not just the delph it was served in. Metonomy is considered semantically regular as there is always a strong association between concepts related in this way. *Synecdoche* is often regarded as a subtype of metonymy. It is based on a part-whole or whole-part relation between the concepts involved; for example, saying 'I'll have the chicken' to refer to a meal which has chicken as a constituent element. Other examples of sense extension, including what we term analogical sense extension, are referred to as *irregular polysemy.*

Lexical Rules Most research on sense extension has taken place within the linguistic framework of the *generative lexicon* developed by Copestake[3, 4]. In the generative lexicon framework, both metynomic and metaphorical (irregular) sense extension are handled by *lexical rules.* These rules are expressive enough to cover a number of forms of sense extension, and allow for productivity within specified parts of the lexicon[5].

One of the suggested rules for sense extension within this framework is called *grinding.* Grinding allows the word for a countable noun object; for example, "an airplane"; to refer in a mass sense to some substance derived from that object; for example, "after the crash, there was airplane all over the field". Another lexical rule that impacts sense extension is what is known as the *blocking* rule, which imposes the constraint that a sense cannot be extended if there already is an existing word covering that extension. For example, by the grinding rule described above we should be able to use the word "pig" to refer to both the animal; for example, "a pig"; and the substance derived from that animal; for example, "I love sweet and sour pig!". However, the fact that there is already an

existing word for the substance derived from pig, "pork", blocks the extension in this case. The dative construction rule is another relevant rule.

The lexical rule account is intended to explain both metynomic and metaphoric sense extension. However, it is difficult to see how such rules can explain the apparent importance of structural matching or similarity between domains in sets of extended senses as seen in the aircraft and seacraft example. We will call this type of sense extension *analogical sense extension*. Lakoff[6] has provided a discussion of sense extension based on metaphor and analogy.

2.2 Analogy

Analogy Analogy is the process of understanding something new in terms of something familiar[7]. The new domain is called the *target analog*, and the familiar domain is called the *base analog*. An analogy *maps* elements from the source domain to elements of the target domain, although elements of both domains may remain unmapped.

A classic example of an analogy is the analogy between the atom and the solar system. There are many reasons for thinking that the atom "is like" the solar system: the electrons orbit the nucleus, and the planets orbit the sun; the mass of the nucleus is greater than the mass of the electrons, and the mass of the sun is greater than the mass of the planets; the positive charge of the nucleus attracts the negatively charged electrons, and gravity attracts the planets to the sun. Certain elements may be put into correspondence between the two domains: electrons and planets; the sun and the nucleus; the sun's gravity and the opposing charges of the electrons and the nucleus.

Analogy may be used as both an aid to memory, and to understanding. Conjectures about unfamiliar domains may also be triggered by analogy. These conjectures are often referred to as *inferences*, or *candidate inferences*.

Structure-Mapping Theory The Structure-Mapping Theory (SMT) of analogy[8] was first described by Gentner in 1983. This theory states that it is the relations between the elements of the target domain and elements of the base domain that provide the most important type of similarity for the mapping in analogy-making. This property of similarity of relations is known as *structural similarity*. A consequence of this is that the properties of the elements of the domains themselves are far less important for the mapping process, as these properties only govern *appearance similarity*. The different types of similarity considered by the SMT are shown in Table 1.

The Structure-Mapping Engine[9] (SME) provides a computational model of the SMT. SME is capable of taking a description of a base and a target domain, and generating conjectures about the target domain by drawing *inferences* from the base domain. For the atom and solar system example, introduced in Section 2.2, SME is presented with a "complete" description of the base analog (the solar system) which includes the fact that the planets revolve around the sun due to the difference between the mass of the sun and the mass of the planets, and because of gravity. SME then produces a series of possible mappings

Table 1. The Different Types of Similarity for Analogical Mapping

Type of Similarity	Important Factor(s) for Mapping
Structural Similarity	Relations between elements.
Literal Similarity	Relations between elements, and properties of elements.
Appearance Similarity	Properties of elements.

between the two domains. Each possible mapping is given a rating based on either the structural, literal, or appearance criteria. The mapping with the highest score is considered the "best" mapping. For this example, the structural similarity mapping produces a candidate inference for explaining the revolution of the electrons around the nucleus in terms of the revolution of the planets around the earth. The candidate inference suggests that it may be because of the difference in mass between the nucleus and the electrons, and because of the mutual attraction between the nucleus and the electrons that the electrons in the atom revolve around the nucleus. The importance of structural, as opposed to appearance, similarity becomes clear in this case; it does not matter that the nucleus is not a giant burning ball of gas, what matters are the relations that hold between the nucleus and the other elements in the atom. This example shows how a more complete understanding of a target domain may be obtained via the process of analogical comparison with a more fully understood base domain.

Analogy and Language The role of analogy in language-learning has recently been studied. Gentner and Namy[10] provided evidence for the argument that language may be acquired through the use of domain-general learning mechanisms (with analogy as one of these mechanisms) rather than through some language-specific acquisition mechanism. Their work, however, uses young children as subjects, and does not investigate how adults learn and extend words. The main finding of the paper is that children initially use perceptual matching when comparing objects if they only have one exemplar, but can use relational matching if they are provided with enough perceptually different exemplars which share conceptual relations. A child presented with a bicycle as an example of a "blicket", and asked to choose as another "blicket" either a pair of glasses (perceptually similar) or a skateboard (conceptually similar), chooses the pair of glasses. A child presented with both a bicycle and a tricycle as examples of "blickets", however, chooses the skateboard. This work suggests a strong connection between the analogical process and language in general. It has also been suggested that analogical mapping contributes to the learning of grammar.

2.3 Questions about Analogy and Sense Extension

Although Lakoff does provide an explanation of sense extension in terms of analogy, we have been unable to locate literature describing experimental work carried out to verify this theory. It seems likely that the extension of meaning

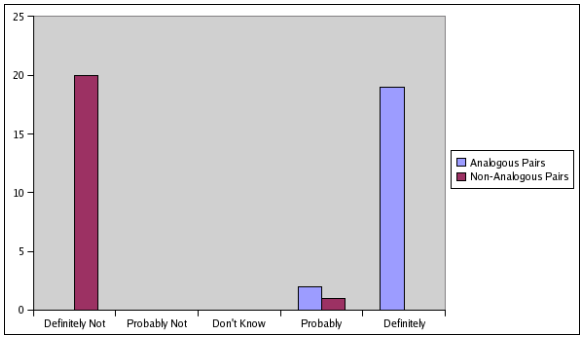


Fig. 1. Results of Pretest Check for Existence of Analogy between Passages

from one domain to another, in some cases at least, depends on the existence of an analogy between the domains. In the experiment described in Section 3 we investigate if such a relationship between analogy and sense extension exists. We are also interested in any other factors which may interact with analogy in this type of sense extension. For example, the blocking rule from the generative lexicon account of sense extension may disallow the extension of a word because a second name for something would most likely be redundant. It is possible that the commonality or generality of words may also influence the ease with which they can be extended.

3 Experiment

The aim of the experiment was to investigate the relationship between analogy and sense extension. A pretest was carried out before the main experiment to check assumptions made in the main experiment. Both the pretest and the main experiment were paper-based.

3.1 Pretest

In the pretest, the participants were presented with six pairs of passages to read. Each passage in the test describes a particular domain. The passages were custom-written for the test, and cover various scientific and technological topics. We had judged there to be an analogy present between the constituent passages for three of the pairs, and no analogy between the passages in the remaining three pairs. The pretest was carried out to check this judgement, and the test involved seven native English-speaking participants.

The order of the pairs of passages was randomly selected for each participant. Participants were asked to read each pair of passages and then indicate if they thought there was an analogy between the domains or not, as shown in the first question in Figure 2. They were then asked to provide details of the mappings in the analogy, as shown in the second question of Figure 2, if they judged there

to be one present. The results of the pretest, as shown in Figure 1, confirmed our initial judgements about the existence of analogies between the pairs of passages. For the analogous pairs, there were 17 "correct" mappings suggested by the participants. Of these, 12 were considered strong mappings as they were suggested by *all* the participants, and 5 were considered weak mappings as they were suggested by only *a subset of* the participants. This distinction between strong and weak mappings was used in the design of the main experiment.

3.2 Main Experiment

Experimental Design The same six pairs of passages used in the pretest were used in the main experiment. All participants were asked to read each pair of passages, and to answer subsequent questions asking them to judge the acceptability of extending words between the domains. There were two groups of participants in the experiment: an analogy group, and a non-analogy group.

In the analogy group, the participants were encouraged to think about possible analogies by giving them the pretest task to perform prior to answering the questions regarding the acceptability of sense extension between the domains. In the non-analogy group, the word "analogy" was not mentioned at all in the experimental materials, and the participants were simply asked to read the passages and answer the questions about sense extension.

The analogy group was primarily useful for investigating the possible relationship between the noting of an analogous mapping and it's subsequent use in sense extension. As the participants in the non-analogy group were not asked to provide details of any analogy they perceived between the passages this analysis was not possible with this group. The two groups also enabled investigation into the possibility of priming for sense extension by drawing the participants' attention to the idea of looking for analogies between the passages.

Participants Twenty native English-speakers participated in the experiment. The participants were not a homogenous group, but were a varied group including teachers, engineers, students, accountants, and architects. The participants were randomly assigned to either the analogy group or the non-analogy group. The experiment was balanced by placing ten people in each group.

Materials Figure 2 shows an example of a pair of passages from the experiment, along with the accompanying questions, as presented to the analogy group in the main experiment. The example pair is the "sewers and veins" example. The first two questions are the pretest tasks, and the remainder are the questions asking about the acceptability of sense extension between the domains of the passages. The first prestest question asked the participants to judge if they thought there was an analogy between the passages on a five-point scale: Definitely Not, Probably Not, Don't Know, Probably, and Definitely. The second

Read the following two short passages carefully, and answer the related questions.

Passage A

In humans, veins are blood vessels that carry blood from organs toward the heart, and arteries are blood vessels that carry blood away from the heart. Systemic veins carry deoxygenated blood containing CO_2 and cellular waste. The pulmonary vein is the exception; it carries highly oxygenated blood from the lungs to the heart. In the lungs oxygen is picked up and CO_2 is given up. As blood pressure is lower in veins than in arteries, veins have venous valves to prevent blood flowing backwards due to other forces; for example, gravity.

Passage B

Ideally a sewer system would be completely gravity-powered. The sewerage pipes from buildings feed into sewer mains. Sewer mains flow into progressively larger pipes until they reach a wastewater treatment plant. This plant is usually located in a low-lying area to aid the flow of wastewater. Often a grinder-pump or a lift-station must be used to move the wastewater; for example, over a hill. Sewers have valves to stop the backflow of waste; for example, floating ball valves and swinging check valves.

Questions

1. Do you think there is an analogy between the subject(s) described in Passage A and the subjects(s) described in Passage B? Circle your answer.
   Definitely Not *Probably Not* *Don't Know* *Probably* *Definitely*

2. If you think there is, or may be, an analogy:
   Please draw lines connecting the analogous elements from the two passages. Each connecting line must connect an element from Passage A to an element from Passage B. Note that not all elements must be connected, and that an element may be connected to multiple other elements.

Systemic Vein	*Wastewater*
Venous valve	*Treatment plant*
Artery	*Floating ball valve*
Pulmonary vein	*Sewer*
Deoxygenated blood	*Grinder-pump*
Heart	*Swinging check valve*
Lungs	*Low-lying area*

3. Which of the following words do you think would be an acceptable alternative name for the *venous valve* in Passage A? Circle your answer.
   Grinder-pump *Floating ball valve* *Sewer* *None of these*

4. Which of the following words do you think would be an acceptable alternative name for the *systemic vein* in Passage A? Circle your answer.
   Grinder-pump *Wastewater* *Sewer* *None of these*

5. Which of the following words do you think would be an acceptable alternative name for the *lungs* in Passage A? Circle your answer.
   Low-lying area *Treatment plant* *Sewer* *None of these*

Fig. 2. Example Pair of (Analogous) Passages for Analogy Group in Main Experiment

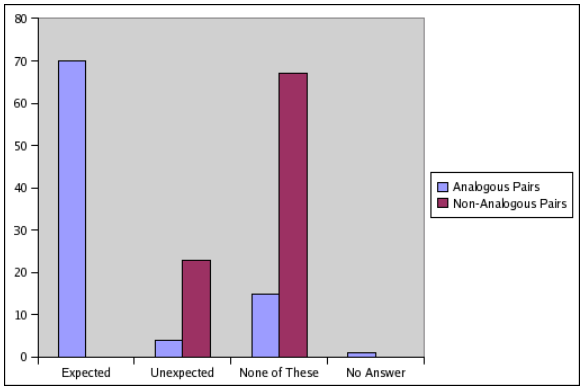


Fig. 3. Non-Analogy Group Results for Sense Extension Questions

pretest questions list some items from both domains and ask the participants to draw lines connecting the analogous elements.

There are three sense extension questions. For the analogous pairs of passages, two of these questions were based on strong mappings, and one was based on a weak mapping (as defined in Section 3.1). Each of the sense extension questions has four possible answers. One of these is the element for the target domain from the "expected mapping" (from the pretest). Two others are items from the target domain that a mapping is not "expected" for given the particular item from the source domain that the question is being asked about. The final option in all cases is "None of These". The order of the expected and unexpected mappings is random in all questions.

For half of the pairs the questions were asked with Passage A as the source, and Passage B as the target, and the other half were the reverse. The participants in each group were given the six pairs of passages in a random order, with the same random order used for both the analogy group and the non-analogy group.

3.3 Results

Non-Analogy Group The results of the experiment for the non-analogy group are shown in Figure 3. A tendency to choose acceptable alternative names based on analogous mappings between the passages in the cases where the pair of passages are analogous is apparent. When the pair of passages are not analogous, participants were most likely to reject all the suggested alternative names in favour of the "None of These" option.

Analogy Group The results of the experiment for the analogy group are shown in Figure 4. These results show a very similar pattern to the results of the non-analogy group in Figure 3. Additional analysis of the data for the group is possible due to the information gathered about the analogies perceived by participants in the pairs of passages. For each sense-extension question per participant

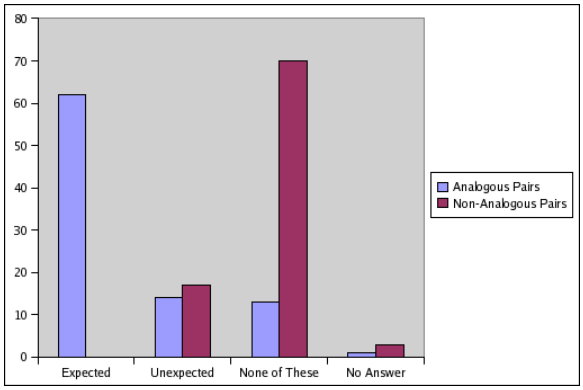


Fig. 4. Analogy Group Results for Sense Extension Questions

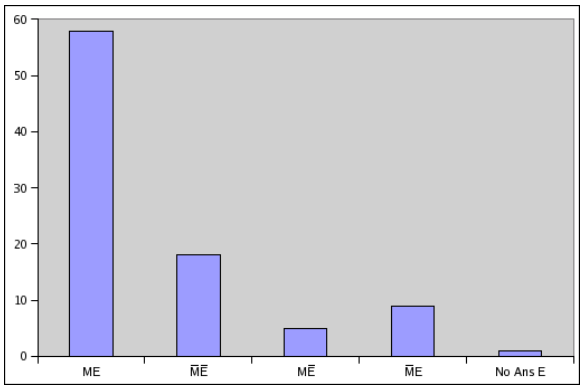


Fig. 5. Analogy Group Results for Expected Mapping and Expected Sense Extension

it was noted if the participant perceived the "expected" mapping, and if they made the "expected" sense extension. Figure 5 shows the results of this analysis for the analogous passages. The "expected" mapping is represented in Figure 5 by M, and an unexpected mapping or no mapping is represented by $\bar{M}$. The "expected" sense extension, based on the "expected" mapping is represented in Figure 5 by E, and an unexpected extension or no extension is represented by $\bar{E}$. It can be seen in Figure 5 that in the cases where participants made the "expected" mapping they tended to use this mapping to extend the sense of the word in question, and when they did not perceive the "expected" mapping they did not make the extension. From these results it seems clear that the analogous mapping is what enables the sense extension to occur in these cases. The correlation coefficient for the data is 0.63, and this indicates a correlation between M and E. (Sign test: $(N+ = 74, N- = 14, p < 0.0001)$)

It can be seen from Figures 3 and 4 that there is no overall evidence of priming as regards sense extension occuring in the analogy group. In fact it looks as if

the opposite is true, but this is misleading as, due to the small sample size of ten people in this group, the difference is in fact not significant. There was no significant difference found between the proportion of "expected" answers given to the questions based on strong mappings and the same proportion given to the questions based on weak mappings.

4 Conclusions

We have described an experiment which shows that analogical sense extension is a reality. In the analogy group, where such data was available, we found a positive correlation of 0.63 between M and E, where M is a binary variable representing the detection of the analogous match, and E is a binary variable representing the extension of the word indicated by the analogous match. This means that generally participants in our experiment extended the sense of words *only* when they had identified the mapping that the extension was based on. These results imply that the analogical process plays an important role in sense extension.

Future work on this topic could include the construction of a computer model capable of identifying analogous domains in a knowledge base, and suggesting sense extensions based on these analogies. The sense extensions could then be rated by people to verify the validity of the predictions made by the model.

In this exploratory work, we failed to find a priming effect when participants were asked to perform analogy-making tasks with the passages before answering the sense extension questions. An extended study, with a larger sample size, would be of great help in investigating this further.

References

1. Apresjan, J.: Regular Polysemy. Linguistics (142) (1973)
2. Nunberg, G.: Transfers of Meaning. Journal of Semantics (**12**(1)) 109–132
3. The Semi-Generative Lexicon: Limits on Lexical Productivity. In: Proceedings of the First International Workshop on Generative Approaches to the Lexicon. (2001)
4. Pustejovsky, J.: The generative lexicon. (1995)
5. Copestake, A., Briscoe, T.: Semi-productive Polysemy and Sense Extension. Journal of Semantics **12** (1995) 15–67
6. Lakoff, G.: The Contemporary Theory of Metaphor. Metaphor and Thought (1993) 202–251
7. Gentner, D., Holyoak, K.J.: Reasoning and Learning by Analogy: Introduction. American Psychologist **52** (1997) 32–34
8. Gentner, D.: Structure-mapping: A Theoretical Framework for Analogy. Cognitive Science **7** (1983)
9. Falkenhainer, B., Forbus, K.D., Gentner, D.: The Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence **41** (1989) 1–63
10. Gentner, D., Namy, L.L.: Analogical Processes in Language Learning. Current Directions in Psychological Science **15**(6) (2006)

Evaluating the Robustness of Collaborative Web Search

Michael P. O'Mahony and Barry Smyth

School of Computer Science and Informatics,
University College Dublin, Ireland
{michael.p.omahony,barry.smyth}@ucd.ie
http://casl.ucd.ie

Abstract. Collaborative web search utilises past search histories in a community of like-minded users to improve the quality of search results. Search results that have been selected by community members for past queries are promoted in response to similar queries that occur in the future. The ISPY system is one example of such a collaborative approach to search. As is the case with all open systems, however, it is difficult to establish the integrity of those who access a system and thus the potential for malicious attack exists. In this paper we investigate the robustness of the ISPY system to attack. In particular, we consider attack scenarios whereby malicious agents seek to promote particular result pages within a community. In addition, we analyse robustness in the context of community homogeneity, and we show that this key characteristic of communities has implications for system robustness.

Key words: Collaborative Web search, personalisation, malicious attack, robustness

1 Introduction

Traditional search engines operate in a "one-size-fits-all" manner and return the same search results for the same query, irrespective of the interests and needs of those individuals using the service. The rationale for collaborative Web search (CWS) lies in the query repetition and selection regularity that exists in the searches of communities of like-minded users. The ISPY system is one example of such a collaborative approach to search [9]. ISPY is a meta-search framework where submitted queries are first passed to base-level search engines (e.g. Google, Yahoo! etc.). The result lists obtained are then adapted and re-ranked by ISPY according to the heuristic that results which have been selected for similar queries in the past are also likely to be relevant for other community members. In previous work, the benefits of this collaborative approach to search has been established [8, 9].

The relatively uniform interests and needs that exist in community domains can also, however, be exploited by malicious agents seeking to manipulate search output. For example, users may seek to promote their own work to other community members in order to enhance their reputation or to achieve financial gain.

In related work, the lack of robustness exhibited by collaborative recommender systems against attack has been highlighted [3, 5, 6]. It has been shown that attacks on these systems are capable of significantly biasing the recommendations that are made for target items. These systems are vulnerable given the open manner in which they operate. Since it is practically impossible to assess the integrity of those who use a system, there is no guarantee that the data inserted into a system's database is an accurate reflection of true user preferences.

In this paper we examine the robustness of the ISPY system against attack. We introduce attack models that are designed to associate target result pages with the query-space of particular communities, with the objective of promoting such pages in the result lists of future search sessions initiated by community members. In addition, we analyse the robustness of collaborative search in terms of community homogeneity. This is a key characteristic of communities and depends on such factors as the maturity of communities, user participation levels and community focus. Some communities will naturally have a tight focus (e.g. communities catering for such niche-interests as military history, model aviation etc.) while others will have a broader focus. By definition, all communities strive for a certain degree of speciality in order to satisfy members' interests and needs and it is clearly important to gain an understanding of the relationship between community focus and robustness against attack.

The paper is organised as follows. Section 2 presents an overview of the ISPY system architecture. Attack models are described in Section 3 and an empirical evaluation of these models is provided in Section 4. We discuss the implications of our findings in Section 5 and conclusions are presented in Section 6.

2 Collaborative Web Search Architecture

The collaborative search technique implemented in the ISPY system is conceived of as a form of meta-search. Each new user query, q_T, is submitted to a set of underlying search engines and their results are combined to form a meta-search result-list, R_M. ISPY then processes R_M to produce a new result-list, R_T, which reflects the learned preferences of a community of like-minded users. This is achieved by recording the page selections made for past search queries, thereby enabling such pages to be promoted when similar queries are submitted in future. (For a more comprehensive description of the ISPY system, refer to [9].)

The key data structure used in the ISPY system is the community *hit-matrix*, H, where H_{ij} represents the number of times that page p_j has been selected as a result for query q_i. Thus, the rows of H correspond to the (unique) queries submitted by community members and maintain an account of all page selections made for each query. Note that no record is maintained of which user selected which page; in effect, the hit-matrix serves as an anonymous account of community preferences.

The similarity between a new query, q_T, and a query, q_i, from the hit-matrix is calculated using the Jaccard index (Equation 1), which measures the term overlap between queries. ISPY selects those rows from the hit-matrix whose

corresponding query has a similarity to q_T that exceeds a specified threshold. The pages associated with these rows are called *promotion candidates*.

$$Sim(q_T, q_i) = \frac{|q_T \cap q_i|}{|q_T \cup q_i|} \tag{1}$$

The relevance of page p_j to query q_i is calculated as the relative number of times that p_j has been selected for q_i; see Equation 2. Further, the relevance of p_j to q_T is a combination of $Relevance(p_j, q_i)$ for all q_i's $(q_1, ..., q_n)$ deemed similar to q_T, as shown in Equation 3, where $Exists(p_j, q_i) = 1$ if $H_{ij} > 0$, and 0 otherwise. Each $Relevance(p_j, q_i)$ is weighted by $Sim(q_T, q_i)$ to discount the relevance of results from less similar queries.

$$Relevance(p_j, q_i) = \frac{H_{ij}}{\sum_{\forall j} H_{ij}} \tag{2}$$

$$WRel(p_j, q_T, q_1, ..., q_n) = \frac{\sum_{i=1...n} Relevance(p_j, q_i) \times Sim(q_T, q_i)}{\sum_{i=1...n} Exists(p_j, q_i) \times Sim(q_T, q_i)} \tag{3}$$

This weighted relevance metric is used to rank-order the promotion candidates. These ranked pages are then listed ahead of the remaining meta-search results, which are themselves ranked according to a standard meta-search scoring metric, to give R_T.

3 Attack Methodology

We assume that the objective of attacks is to promote a single target page within a particular ISPY community. For example, consider the author of a Web page related to the Java programming language, who seeks to promote his page above others for all relevant search sessions. In order to ensure successful promotions in the context of the ISPY system, the key requirement for attackers is to associate the target page in the system's hit-matrix with queries that are representative of community search activity. We refer to such queries as *attack queries*.

To illustrate the above process, consider an attack that seeks to promote a particular target page, p_T. Once registered with an ISPY community, attackers can insert any number of attack queries into the community hit-matrix by simply selecting p_T from the result-lists returned from multiple search sessions. Each search session can involve different combinations of query terms (assuming that the target page is indexed by such terms by the underlying search engines), thereby permitting the target page to be associated with a diverse range of queries within the hit-matrix.

By definition, given the focused nature of typical communities, the selection of suitable terms with which to form attack queries should not pose a significant challenge. For example, queries formed using terms such as *php*, *mysql*, *C++* etc. would be good candidates in the case of a technology-related community. By using such popular or frequently occurring terms, high similarities

(Equation 1) between new search queries submitted by community members and attack queries can be achieved, thereby increasing the weighted relevance and ranking (Equation 3) of the associated target page in the search results.

Note that the effort involved in implementing attacks is unlikely to be high, since the insertion of queries into the ISPY system can be readily automated, thereby enabling a large number of such insertions using many combinations of query terms to be made. Further, the fact that ISPY maintains an anonymous database of community preferences is beneficial for attackers since checks on individual user activity are consequently not possible. Thus, in this paper, we consider attack *cost* solely as a function of the number of queries that are inserted in the course of an attack, and the number of terms that are present in each.

We propose two attack models which are described in the following sections. In each case, we assume that full knowledge concerning the distribution of query term usage in a community is known to attackers. While this assumption is favourable from the attacker's perspective, search engines often provide such information (to some degree) to users. For example, the popular search queries that relate to particular query terms are listed on result pages obtained using the Google search engine.

3.1 *Promote-Always* Attack

This attack model seeks to promote a particular target page for all searches submitted by community members, irrespective of the actual search terms that are used. Define attack query size, or the number of terms included in attack queries, as l. In order to ensure maximal coverage of the community query-space, the most frequently-used search terms are selected for attack queries. If multiple attack queries are inserted, each consists of a different combination of query terms. The first attack query consists of those l terms with the highest combined frequency of occurrence, the second attack query consists of those l terms with the second-highest combined frequency of occurrence, etc. For example, if *java*, *oracle* and *linux* were the three most frequently occurring query terms (in descending order) in a particular community, the first two attack queries of size $l = 2$ would be {*java*, *oracle*} and {*java*, *linux*}.

3.2 *Term-Specific* Attack

The objective of this attack is to promote a particular target page when a specific term is used in search queries. Attack queries are comprised of the specific term in question, along with combinations of $l - 1$ other terms which are selected so as to optimally cover the query-space as described above.

4 Evaluation

In this section, we evaluate the attack models described above. We begin by providing details on the configuration parameters used in ISPY, the evaluation dataset and the experimental methodology and metrics used in the analysis.

4.1 ISPY Configuration

ISPY was configured to display a maximum of 10 search results on each results screen, of which at most 3 were promoted results. For example, if the system was able to make 5 promotions for a particular search query, the top-3 promotions would be displayed at the top of the first screen and the remaining 2 would be displayed at the top of the second screen.

ISPY's query-similarity threshold was set to 0.5, so that only those past queries that shared more than 50% of query terms with the current query would be considered to be similar for the purposes of result promotion (see Section 2). The motivation for this relatively high threshold value is to avoid spurious matches between unrelated queries; for example the queries {*jaguar*, *automobiles*} and {*jaguar*, *wildlife*} have a similarity of 0.33 but are clearly unrelated.

4.2 Dataset

The dataset used in our evaluations was obtained from a trial of the ISPY system which took place over an extended period among the 50 staff members of a local software company. ISPY was configured to draw on Google and HotBot as a source of search results. A new community hit-matrix was created for participants, which was trained on search log data prior to the start of the trial.

The dataset consists of 4,744 queries, of which 2,256 were unique, and associated result pages (URL's), of which 3,587 were unique. The total number of unique query terms in the dataset was 2,673.

4.3 Metrics & Methodology

For both attack models, the objective was to promote a new target page which was not already present in the community hit-matrix. Consequently, no promotions were possible for the target page pre-attack. We evaluate the effectiveness of attacks using the *hit ratio* metric [5], which measures the percentage of times that a target page appears in top-N promotions. Unless stated otherwise, we set $N = 3$ in our evaluations since, from an attacker's perspective, the promotion of the target page to the first screen of search results represents optimal attack success.

The following experimental methodology was employed. For the *Promote-Always* attack, test and training sets consisting of randomly selected percentage of queries and associated result selections were drawn from the dataset. Attack queries were added to the training set and the average hit ratio for the target page was calculated over all test set queries. Test set size was 10% and a 10-fold cross validation was performed.

In the case of the *Term-Specific* attack involving a particular specific term, t, the test set consisted of all dataset queries that contained term t. These test set queries were submitted in turn to the full dataset (with attack queries inserted), and the average hit ratio for the target page was calculated. This procedure was then repeated for all the unique query terms contained in the dataset, and the average hit ratio over all such terms was calculated.

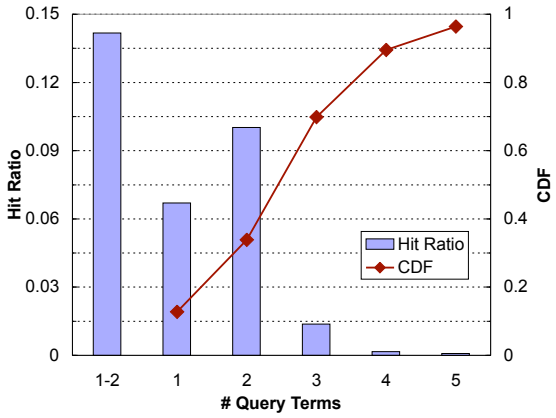


Fig. 1. Hit ratios achieved for the *Term-Specific* attack model versus attack query size. The cumulative distribution function (CDF) of genuine query sizes is also shown.

4.4 Results – Attack Models

We first examine the effect of attack query size on robustness. Results are shown in Figure 1 for the *Term-Specific* attack, for a fixed quantity (12) of attack queries inserted. The highest hit ratios were achieved for query sizes of $l = 1$ or $l = 2$ terms, with the greatest effect observed when a 50%-50% combination of these query sizes was used (labeled "1-2" in the figure). Attack queries formed using 3 or more terms had little effect. The cumulative distribution function of genuine query sizes is also shown in Figure 1, which shows that 70% of queries consisted of 3 terms or less. (Genuine queries consisted of 3 terms on average, with a standard deviation of 1.3.) Given the relatively stringent overlap criteria imposed by the high query-similarity threshold value of 0.5, attack queries consisting of smaller combinations of frequently occurring terms had a greater probability of exceeding the similarity threshold with genuine queries. A similar result was found for the *Promote-Always* attack. In the remainder of this section, attacks are implemented using the combination of query sizes discussed above.

We now compare the performance of the two attack models at various attack sizes (expressed as a percentage of the number of attack queries inserted to the total number of genuine queries present in the dataset). In the case of the *Promote-Always*, a hit ratio of only 7% (Figure 2) was achieved when the largest quantity (20%) of attack queries was inserted, which indicates that ISPY proved to be quite robust to this form of attack. Given the frequency of occurrence of query terms in the dataset follows a Zipfian distribution (i.e. only a small number of terms appear frequently in search queries; see Figure 3) and the high query-similarity threshold used, the difficulty of creating attack queries that effectively span the entire query-space becomes apparent. The robustness of ISPY is further enhanced by limiting the number of promotions to be displayed on the first results screen to 3, thereby providing an additional challenge for attackers.

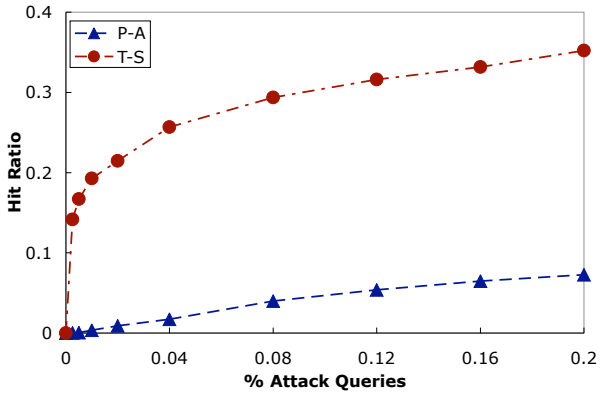


Fig. 2. Hit ratios achieved for the *Promote-Always* (P-A) and *Term-Specific* (T-S) attack models versus the quantity of attack queries inserted.

In contrast, the *Term-Specific* attack performed significantly better, achieving hit ratios of 14% at the smallest attack size (0.25%, or 12 attack queries inserted) and 35% at the largest attack size (20%). The improved performance of this attack model was to be expected, given its more focused nature, where promotions were attempted only in cases where a particular target term was used in search sessions. This attack was also very straightforward to implement since 50% of the attack queries created consisted of only 1 term – the target term, with the remaining 50% consisting of combinations of the target term plus only a single additional term. Thus, we can conclude that ISPY was indeed vulnerable to this particular form of attack, given that small numbers of easily-created attack queries were capable of significantly biasing the output of the system. In addition, these findings have significance from the perspective of attack detection, where the correct classification of relatively small quantities of attack data is likely to prove problematic.

4.5 Community Homogeneity

Different communities have varying degrees of homogeneity, i.e. the degree of regularity and repetition that exists in the searches of like-minded community members. A key measure of homogeneity is the distribution of search terms in the query-space. As mentioned above, the distribution of the lowest 500-ranked query terms (the most significant part of the distribution since lower-rank query terms have the highest frequency of occurrence) in the dataset is Zipfian, with an exponent of $\approx$ -0.65 (Figure 3). While we do not have data from other ISPY communities with different degrees of homogeneity, it is, however, possible to simulate community homogeneity by varying the distribution of query term frequency of occurrence.

In the following experiments, we maintained the same number of queries and result pages (URL's) as before, and randomly assigned terms to queries

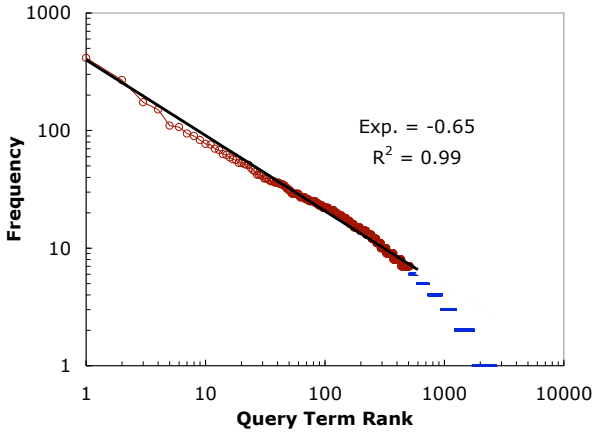


Fig. 3. The frequency of occurrence of query terms in the dataset. The distribution of the lower-ranked query terms is Zipfian, with an exponent of ≈ -0.65

according to Zipfian distributions. We conducted experiments with exponents ranging in values from -0.2 to -1.4. Exponents closer to zero resulted in more uniform distributions of query terms (i.e. more heterogeneous communities).

Results are shown in Figure 4 for *Promote-Always* attacks for attack sizes of 10%, which were implemented as described previously. It is clear that robustness deteriorated substantially as community homogeneity increased. For example, a hit ratio of 5.8% was achieved for an exponent of -0.6, compared to a hit ratio of 12.7% for an exponent of -1.0 (Top-3). The deterioration in robustness was also evident at higher top-N values. Robustness began to improve for exponents $<$ -1, which is explained by the large increase in the total number of promotions that ISPY was able to make for test queries. For example, 77 and 273 promotions were made for exponents of -1.2 and -1.4, respectively, compared to 19 promotions made for an exponent of -1. Consequently, it became much more difficult for target pages to be promoted into one of the top-N rankings at the highest exponent values.

Note that, in practice, the task of identifying query terms that effectively span the query-space in more homogeneous communities is likely to be straightforward, with the result that attacks against such communities become both easier and less costly, in the sense that fewer attack queries are required to cover the query-space, to implement.

5 Discussion

In this paper we have examined the robustness of the one particular approach to collaborative Web search. In related work [1], the robustness of several navigation-orientated web personalisation algorithms was investigated. This research indicates that some commonly-used algorithms were vulnerable to attacks created

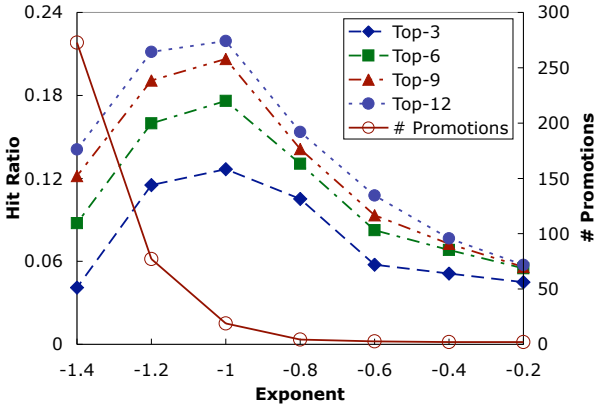


Fig. 4. Hit ratios and the number of promotions made by the ISPY search engine when subjected to a *Promote-Always* attack versus query term distribution.

by generating false clickstreams designed to promote particular target pages. Further, the lack of robustness exhibited by collaborative recommender systems against attack has been well-highlighted in the literature [3, 6]. Recently, an analysis of tagging systems [2] has found that these systems are also prone to manipulation by malicious agents.

All of the above systems rely on data that is gathered explicitly or implicitly from users in order to make recommendations or to facilitate search and navigation. Given this open manner of operation, these systems are readily susceptible to manipulation. Users are unlikely to remain loyal if it is perceived that recommendations of suspect quality are being made and thus, the integrity of these systems is of prime importance. In this regard, recent research which proposes techniques to improve the robustness of collaborative recommender systems against attack [4, 7], is of interest. These and other techniques provide a useful basis for future work on attack detection in related systems.

6 Conclusions

Collaborative Web search leverages the search histories of communities of like-minded users to adapt and re-rank search results to satisfy the needs of community members. Results that have been selected for queries in the past are promoted when similar queries are submitted in future search sessions.

The repetitive nature of community search activity can also, however, be usefully exploited by malicious agents in order to to bias search output. In this paper, we have introduced attack models which seek to promote target pages within communities by linking such pages with frequently-used search terms. Our findings, based on real-world community data collected from a local software company, indicate that ISPY exhibited a significant degree of robustness against

the more general *Promote-Always* attack model, but was susceptible to low-cost *Term-Specific* attacks consisting of small numbers of attack queries.

In addition, we analysed the robustness of collaborative search in terms of community homogeneity. Different communities will naturally vary in focus, and we have shown, through simulation, that more specialised communities proved significantly more vulnerable to attack. Clearly, it would be useful, if data were available, to establish the range of homogeneity and robustness exhibited by diverse, real-world, communities. One approach we are currently investigating is to perform a cluster analysis of our existing dataset to discover groups of related queries, and thereby obtain sub-communities with different degrees of focus.

In future work we also will expand our robustness analysis to other adaptive systems. In addition, we will investigate techniques that provide for the effective detection and elimination of attack data.

Acknowledgments. This work was carried out with the support of Science Foundation Ireland under Grant No. 03/IN.3/I361, which is gratefully acknowledged. We would also like to thank our reviewers for their valuable comments.

References

1. R. Bhaumik, R. Burke, and B. Mobasher. Effectiveness of crawling attacks against web-based recommender systems. *In Proceedings of the 5th Workshop on Intelligent Techniques for Web Personalization (ITWP-07)*, (To Appear).
2. G. Koutrika, F. A. Effendi, Z. Gyongyi, P. Haymann, and H. Garcia-Molina. Combating spam in tagging systems. *In Proceedings of 3rd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb'07)*, May 8 2007.
3. S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. *In Proceedings of the 13th International World Wide Web Conference*, pages 393–402, May 17–20 2004.
4. B. Mehta, T. Hofmann, and P. Frankhauser. Lies and propaganda: Detecting spam users in collaborative filtering. *In Proceedings of the 12th Internation Conference on Intelligent User Interfaces (IUI-07)*, pages 14–21, January 28–31 2007.
5. B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering system. *In Proceedings of the 2005 WebKDD Workshop (KDD'2005)*, 2005.
6. M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Recommender systems: Attack types and strategies. *In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 334–339, July 9–13 2005.
7. M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Detecting noise in recommender system databases. *In Proceedings of the International Conference on Intelligent User Interfaces (IUI'06)*, pages 109–115, Jan 29–Feb 1 2006.
8. B. Smyth, E. Balfe, O. Boydell, K. Bradley, P. Briggs, M. Coyle, and J. Freyne. A live-user evaluation of collaborative web search. *In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1419–1424, 2005.
9. B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell. Exploiting query repetition & regularity in an adaptive community-based web search engine. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 14:383–423, 2004.

Finding the Most Satisfiable Maximal Relaxation in Over-Constrained Problems

Alexandre Papadopoulos and Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{a.papadopoulos|b.osullivan}@4c.ucc.ie

Abstract. We consider the problem of generating maximal relaxations in the context of large real-world configuration problems. We focus on an automaton-based approach because it is becoming standard in real-world configuration. Two novel algorithms are presented. The first algorithm finds from amongst the longest relaxations to a set of inconsistent user constraints, the one that is consistent with the largest number of solutions; this algorithm is linear in the size of the automaton. The second algorithm finds from amongst the set of maximal relaxations, the one that is consistent with the largest number of solutions. While this algorithm is not polynomial in the size of the automaton, we show that in the average it is more than 500 times faster than a current state-of-the-art algorithm. Results are reported on a large configuration problem from the automotive industry.

1 Introduction

We consider a configuration tool where a user can specify preferences for options. These preferences are expressed as constraints. When preferences conflict, we want to help the user find which preferences to relax. In an iterative process, the user might relax constraints until at least one consistent solution is found. Alternatively, the user might wish to be told which particular subsets of his constraints can be satisfied. Most current approaches to explanation generation in constraint-based settings are based on the notion of a (set-wise) minimal set of unsatisfiable constraints, also known as a minimal conflict set of constraints. To demonstrate the concepts, we provide a simple example.

Example 1 (Car Configuration). Consider a simple car configuration problem, based on an example in [8], with the following set of options; the Boolean variable $x_i \in \{0, 1\}$ indicates whether constraint c_i is in the current set of active constraints or not:

Constraint	Option	Selector	Cost
c_1	Budget	$x_1 = 1$	$\sum_{i \in \{2,\ldots,5\}} (k_i \cdot x_i) \leq 3000$
c_2	Roof Rack	$x_2 = 1$	$k_2 = 500$
c_3	Convertible	$x_3 = 1$	$k_3 = 500$
c_4	CD Player	$x_4 = 1$	$k_4 = 500$
c_5	Leather Seats	$x_5 = 1$	$k_5 = 2600$

Assume that the technical constraints of the configuration problem forbid convertible cars having roof racks, therefore, constraints c_2 and c_3 form a conflict. Note that, given the budget constraint, if the user selects option c_5, it is not possible to have any of the options c_2, c_3, c_4. ▲

The set of all minimal conflicts for this example are: $\{c_2, c_3\}, \{c_1, c_2, c_5\}, \{c_1, c_3, c_5\}$, and $\{c_1, c_4, c_5\}$. As explanations, these conflicts are sufficient to explain, using a subset of the user's constraints, why all constraints cannot be satisfied simultaneously. Based on the set of minimal conflicts we can compute the set of set-wise maximal relaxations showing which of the user's constraints can be satisfied. Table 1 presents the set of all maximal relaxations, each showing how the user can satisfy at least some of his constraints. For example, consider Explanation I: we can simultaneously satisfy the constraints in $\{c_3, c_4, c_5\}$, but we must exclude c_1 and c_2.

Table 1. The maximal relaxations and minimal exclusion sets for the over-constrained problem presented in Example 1. We show both the subset of the constraints in the relaxation (marked with a ✓) and those in the exclusion set, i.e. those that must be removed (marked with a ×).

	Constraints						
Exp.	c_1	c_2	c_3	c_4	c_5	Relaxation	Exclusion Set
I	×	×	✓	✓	✓	$\{c_3, c_4, c_5\}$	$\{c_1, c_2\}$
II	×	✓	×	✓	✓	$\{c_2, c_4, c_5\}$	$\{c_1, c_3\}$
III	✓	×	✓	✓	×	$\{c_1, c_3, c_4\}$	$\{c_2, c_5\}$
IV	✓	✓	×	✓	×	$\{c_1, c_2, c_4\}$	$\{c_3, c_5\}$
V	✓	×	×	×	✓	$\{c_1, c_5\}$	$\{c_2, c_3, c_4\}$

In some applications we may have to choose a single relaxation to present to the user. The question is, which one should we select? The obvious response would be to select the relaxation that cannot be extended using any of the user's choices without eliminating all solutions – this is the standard notion of a maximal relaxation. Amongst the set of maximal relaxations we might prefer to select the one that is longest on the basis that it includes the largest number of user constraints. However, an alternative is to present the relaxation that is consistent with the highest number of solutions to the problem, while remaining maximal. This is the question we address in this paper.

We consider the problem of generating the most soluble maximal relaxation in the context of large real-world configuration problems. As a running example, we consider an industrial sized configuration problem, based on the Renault Mégane car. In Section 2 we summarise the preliminary concepts required in this paper and present results from a motivating experiment showing that the number of solutions consistent with a maximal relaxation is not correlated with its length. We also present some theoretical results on the properties of maximal relaxations. In Section 3 we summarise the basics of automaton-based configuration. We focus on an automaton-based approach because it is becoming standard in practice. In Section 4 we present two novel algorithms. The first algorithm finds from amongst the *longest relaxations* to a set of inconsistent user constraints, the one that is consistent with the largest number of solutions; this algorithm is *linear in the size of the automaton*. The second algorithm finds from amongst the set of *maximal relaxations*, the one that is consistent with the largest number of solutions. While this algorithm is not polynomial in the size of the automaton, we show that in the average it is more than *500 times faster than a state-of-the-art algorithm*.

Our experiments are summarised in Section 5. We discuss related work in Section 6, and conclude in Section 7.

2 Preliminaries on Maximal Relaxation

We focus on constraint satisfaction problems in this paper, but the results hold for many other settings in which consistency is monotonic. This property holds whenever the set of solutions to a set of constraints $\mathcal{C}$ is a subset of the solutions to any set of constraints that is a subset of $\mathcal{C}$.

In addition, we focus on constraint satisfaction problems that are solved in an interactive manner, e.g. product configuration problems. It is useful to distinguish between a background set of constraints, $\mathcal{B}$, that cannot be relaxed, and a set of constraints, $\mathcal{U}$, that are added by the user as he finds a preferred solution to $\mathcal{B}$ by finding a solution to $\mathcal{B} \cup \mathcal{U}$, the constraint problem we denote as $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{B}, \mathcal{U} \rangle$.

A set of constraints is consistent if it admits a solution. We will assume that the set of background constraints, $\mathcal{B}$, admits at least one solution. If a set of constraints does not admit a solution, at least one constraint must be excluded in order to recover consistency. Specifically, we are interested in finding *maximal relaxations* of $\mathcal{P}$.

Definition 1 (Maximal Relaxation). *Given a constraint problem $\mathcal{P} \stackrel{\text{def}}{=} \langle \mathcal{B}, \mathcal{U} \rangle$ that is inconsistent, a subset R of $\mathcal{U}$ is a* relaxation *of $\mathcal{P}$ if $\mathcal{B} \cup R$ admits a solution. The relaxation R is a* maximal relaxation *if $\forall R' \supset R$, $\mathcal{B} \cup R'$ is inconsistent.*

While intuitively we might believe that longer relaxations have fewer solutions, the story is not so simple. In Figure 1 we show the results of a simple experiment on the Renault Megane configuration problem [1], which has been compiled in an automaton. This problem has 99 variables and about 2.8×10^{12} solutions. We built inconsistent user queries that instantiated 40 randomly chosen variables with a random value. We ran 20 such queries. For each query, we generated the complete set of maximal relaxations using the Dualize & Advance algorithm [2]. Using the automaton we could efficiently count the number of solutions consistent with each relaxation. In Figure 1 we plot, for each maximal relaxation, its length and the number of solutions of the problem consistent with it. It is clear from this figure that the number of solutions of a maximal relaxation is not necessarily correlated with its length.

Of course, theoretically, there is no reason why the number of solutions of two maximal relaxations should be similar. The following example shows they can be arbitrarily different. We have the variables $x_0, \ldots, x_n$ with respective domains $D(x_0) = \{0, 1\}$ and $D(x_i) = \{0, \ldots, d\}, i > 0$, and the constraints $x_0 < x_i, \forall i > 0$, $x_0 > x_i, \forall i > 0$ and $x_0 = 0$. This problem is inconsistent, and $R_1 = \{x_0 < x_i, \forall i > 0\} \cup \{x_0 = 0\}$ and $R_2 = \{x_0 > x_i, \forall i > 0\}$ are two maximal relaxations of the constraints. The number of solutions to R_1 is d^n, while there is only one solution to R_2.

The problem of selecting the maximal relaxation consistent with the largest set of solutions is intractable, in general. The basic decision problem involved is as follows:

> Given an inconsistent set U of constraints and a maximal relaxation $R \subseteq U$, does another maximal relaxation $R' \subseteq U$, $R' \neq R$, exist such that the number of solutions of R' is greater than the number of solutions of R?

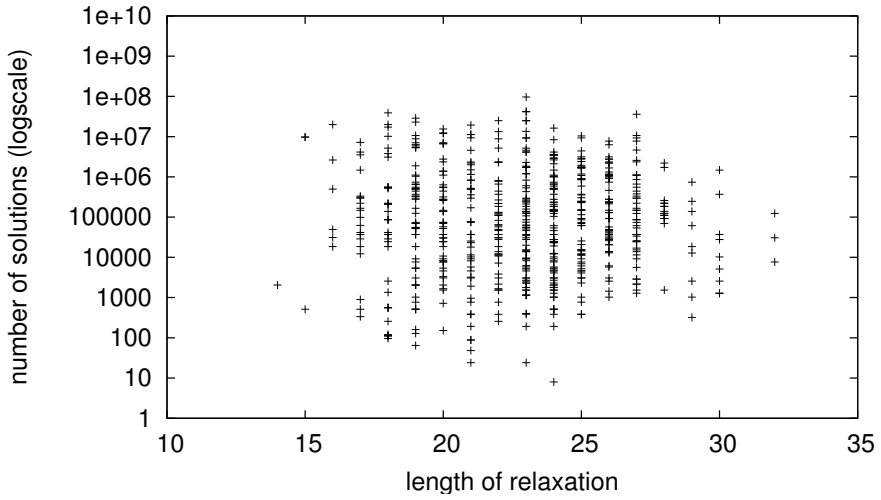


Fig. 1. Results from a simple experiment showing that number of solutions of a maximal relaxation is not necessarily correlated with its length.

Although no formal proof of the complexity of this problem has been established, this problem is highly combinatorial; consider that the number of maximal relaxations is exponential in the number of constraints, and counting the number of solutions to a maximal relaxation is $\#\mathcal{P}$-Complete. However, we will present an algorithm that can find the maximal relaxation consistent with the largest set of solutions very quickly, in practice, for interactive applications.

3 The Basics of Automaton-based Configuration

In a configuration context, a typical approach is to compile the problem into an automaton in order to facilitate interactive solving [1, 15]. In this case many operations become tractable in practice. Let us focus, therefore, on the case where we have a compiled form of a problem, and the user chooses only unary constraints, i.e. assignments or disjunctions of assignments to the variables. A query will be composed of a set of constraints, each constraint c_i of which holds on a variable x_i.

An automaton gives a compact way of representing the set of solutions to a problem. Informally, an automaton can be seen as a representation of the search tree on which minimisation allows us to reduce its size. This automaton only recognises words of the same length, and each recognised word corresponds to a solution of the problem, a particular ordering on the variables having been fixed in advance.

The incoming and outgoing transitions of a state q are denoted by $in(q)$ and $out(q)$, respectively. The origin and destination state of a transition t are denoted by $in(t)$ and $out(t)$, respectively. The initial and the final states (or the source and the sink) are denoted by I and F, respectively. The level of a state q is the length of the words from I to q. The set of all states of level i is noted $Q(i)$. The level of a transition t is the level of

$out(t)$. Each level greater than 0 corresponds to a variable of the problem. Thus, each transition t provides a support for the instantiation of the variable of its level with the value labelling t.

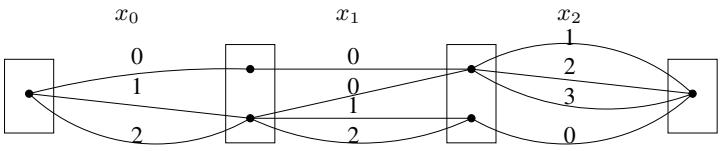


Fig. 2. An example automaton defined on three variables.

Figure 2, for example, shows the automaton for a problem on three variables x_0, x_1 and x_2. This problem has 13 solutions, some of which are $001, 002, 102$, etc.

4 Algorithms

We now present two novel algorithms for finding relaxations that are consistent with the largest number of solutions, based on an automaton representation of the configuration problem. For a particular user query, comprising a set of unary constraints that restrict the domain of each variable, a valuation $\phi(t)$ is associated with each transition t of the automaton: $\phi(t) = 0$ meaning that this transition supports a valid instantiation (i.e. is labelled by an allowed value) and $\phi(t) > 0$ meaning it does not. Thus, to each complete path from I to F there corresponds a relaxation of the user's constraints, composed of the user's constraints supported by the transitions of the path with a valuation of 0.

If we restrict the valuation of the transitions only to 1 in case of a violation, the cost of a path from the source to the sink, which is the sum of the valuations of the transitions it is composed of, corresponds to the number of user constraints violated. If no such path of cost 0 exists, then the set of user constraints is inconsistent. A procedure is described in [1] that associates with each transition t of an automaton a cost $cost(t)$ of the best path (i.e. of minimal cost) of the automaton that uses t. This allows us to explore only the shortest paths in the automaton and, thus, only the longest relaxations of the user's constraints. Therefore, this can give our first exact algorithm (Algorithm 1) that finds, amongst all the longest relaxations, the one that is consistent with the largest number of solutions, in time linear in the size of the automaton.

This algorithm works by associating with each state q' of the automaton, the most soluble longest relaxation restricted from I to q' (the automaton "to the left of" q'), stored in $relax(q')$, with $nsols(q')$ storing the corresponding number of solutions. The set $candidates$, after the iteration (starting at line 6), will contain all the longest relaxations ending at q'. At this point, we can choose the most soluble relaxation, as a less soluble one could not result in a longer relaxation globally more soluble. As the size of $candidates$ is bounded by the number of states of the previous level, this algorithm runs in time linear in the size of the automaton.

Algorithm 1: Finding a most soluble longest relaxation.

Data: An automaton updated for a user query.
Result: A most soluble longest relaxation.

```
1  relax(I) ← ∅
2  nsols(I) ← 1
3  for i ← 1 to n do
4      forall q' ∈ Q(i) do
5          candidates ← ∅
6          forall t ∈ in(q') s.t. t is optimal do
7              q ← in(t)
8              if φ(t) = 0 then R ← relax(q) ∪ {c_i}
9              else R ← relax(q)
10             candidates ← candidates ∪ {R}
11             nsols(R) ← nsols(R) + nsols(q)
12         nsols(q') ← max_{R∈candidates} nsols(R)
13         relax(q') ← R ∈ candidates s.t. nsols(R) = nsols(q')
14 return relax(F)
```

However, restricting to the longest relaxation can prove to be too strong. For example, the plot in Section 2 (Figure 1) suggests that there is quite a concentration of long maximal relaxations, but very few of maximum length. Focusing on candidates amongst the maximal (by inclusion) relaxations seems to be a good trade-off between solubility and maximality. Therefore, we can adapt the previous algorithm to explore the whole automaton so as to consider all relaxations. The difference is that we cannot now greedily keep partial optimal solutions, because what is locally a maximal relaxation may not eventually be maximal.

For example, in the automaton of Figure 2, suppose we want every variable to be 0. Two maximal relaxations start in the second state of level 1: c_1 and c_2. The first has 3 solutions while the second has only 2. But the first will be, at the next step, included in the relaxation c_0c_1, which has 3 solutions, while c_2 will still be a maximal relaxation, but with 4 solutions. Therefore, we need to maintain for each state the list of all the maximal relaxations. This procedure has, therefore, a complexity linear in the size of the automaton times the number of maximal relaxations.

The corresponding modification is presented in Algorithm 2. This is essentially an ad-hoc procedure that lists all the maximal relaxations of a query. However, being specifically designed for our context, it can be more efficient than generic algorithms, such as Dualize & Advance [2], and gives, at the same time, the number of solutions of each relaxation. In this algorithm $relax(q)$ is now a set of relaxations, and for each of them, say R, $nsols(q, R)$ stores its corresponding number of solutions. At lines 12 and 13, any relaxation that is a subset of another is removed, so as to keep only the maximal elements. As the size of the list $relax(q)$ is bounded by the total number of relaxations, the complexity is linear in the size of the automaton times the number of relaxations.

Algorithm 2: Finding a most soluble maximal relaxation.

Data: An automaton updated for a user query.
Result: A most soluble maximal relaxation.

```
1  relax(I) ← {∅}
2  nsols(I, ∅) ← 1
3  for i ← 1 to n do
4      forall q' ∈ Q(i) do
5          forall t ∈ in(q') do
6              q ← in(t)
7              forall R ∈ relax(q) do
8                  if φ(t) = 0 then R' ← R ∪ {c_i}
9                  else R' ← R
10                 relax(q') ← relax(q') ∪ {R'}
11                 nsols(q', R') ← nsols(q', R') + nsols(q, R)
12         Sort relax(q') by decreasing cardinality
13         forall R ∈ relax(q') do Remove in relax(q') subsets of R
14 return relax(F)
```

5 Empirical Evaluation

The objective of our experimental evaluation was to demonstrate the effectiveness of Algorithm 2 against a state-of-the-art algorithm for enumerating all maximal relaxations, and a number of obvious heuristic methods. We did not evaluate Algorithm 1 since it is an exact algorithm, linear in the size of the automaton.

As mentioned in Section 2, we based our experiments on the Renault Mégane Problem, also introduced in [1], which was compiled to an automaton. This problem has 99 variables and over 2.8×10^{12} solutions. Based on this benchmark, we built inconsistent user queries that instantiated 40 randomly chosen variables with a random value. We ran 20 such queries. For each query, we generated the complete set of relaxations using the state-of-the-art Dualize & Advance algorithm [2], for finding all maximal relaxations in a constraint satisfaction context, and compared its performance against that of Algorithm 2 from this paper. For both algorithms, we recorded the time each required to find the most satisfiable maximal relaxation.

In addition to comparing Algorithm 2 against Dualize & Advance, we compared three heuristic techniques in terms of the number of solutions of the best relaxation they found. These heuristic methods were:

1. We considered two heuristics for deciding which user constraints to add first in order to compute a maximal relaxation based on standard search ordering heuristics. The objective of standard variable orderings is to fail as early as possible, so we considered their anti-heuristics. The anti-heuristics we considered were: choose the next assignment whose pre-assignment domain was largest (`max-dom`); and choose the next assignment with largest ratio of its pre-assignment domain divided by the number of constraints connected to this variable (`max-(dom/deg)`).

2. The third heuristic chose as its next assignment the one that would reduce the number of solutions of the remaining problem by the least amount (`minimise solution loss`).

The key measurements taken in each case were the number of solutions of the relaxation found as well as the time taken to find that relaxation.

In Figure 3 we compare each method in terms of the solubility of the best relaxation each found based on each query; note that we sorted the queries by the solubility of the most satisfiable relaxation for the purposes of clarity. We observe that, with the exception of the heuristic that minimises solution loss at each step, the other heuristics performed very poorly, because they never found the best relaxation, and often selected a significantly sub-optimal one.

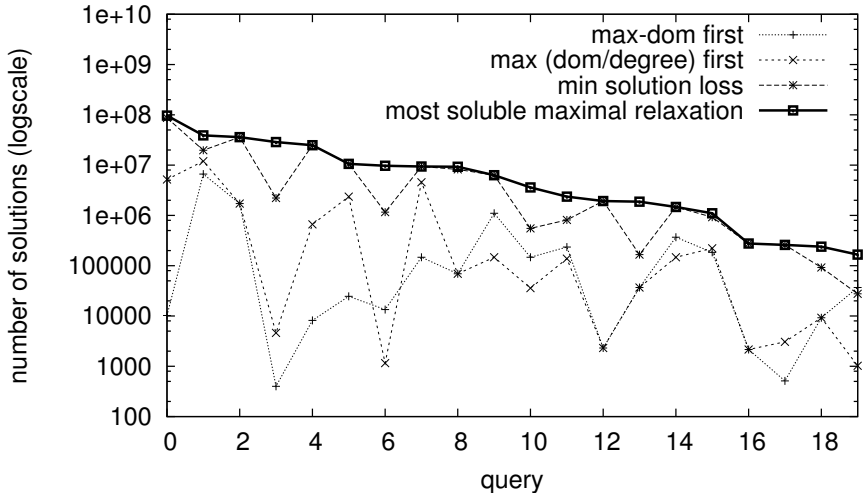


Fig. 3. Solubility of the best relaxation found by each method.

Concerning running time, the time taken to find one relaxation using the `max-dom` and `max-(dom/deg)` heuristics is very fast, but they were not effective at finding optimally soluble relaxations. Therefore, we do not present time results for these heuristics. On the other hand, using the heuristic that minimises the solution loss, `minimise solution loss,` was prohibitively slow. We, therefore, do not discuss time results for this option either.

The most interesting comparison regarding time is between the Dualize & Advance algorithm and our exact algorithm (Algorithm 2) for finding the most soluble maximal relaxation. Our results, summarised in Table 2, show the obvious advantage of our algorithm. Not only does our algorithm guarantee that it will find the maximal relaxation consistent with the most solutions of the problem, it is over 500 times faster than a current-state-of-the-art algorithm. Also, the minimum/maximum and average time re-

Table 2. Running times in seconds for both algorithms

Algorithm	times (seconds) minimum	maximum	average
Dualize and Advance [2]	255	726	416
Most soluble maximal relaxation (Algorithm 2)	0.4	1.3	0.8

quired by Algorithm 2, is of the order of one second, which is ideal for interactive applications.

6 Related Work

There have been many technical papers about explanation in the context of constraints [1, 3, 5, 6, 8, 11, 13, 14]. The dominant approach to explanation in configuration is based on computing minimal conflicting sets of constraints, which is related to the problem of finding all maximal relaxations. Approaches to finding the most preferred relaxations are well-known [8]; ILOG's Configurator product has an explanation generation algorithm based on finding preferred conflicts/relaxations. However, very little has been said about how to choose from amongst the set of all explanations, or how to select amongst equally preferred explanations. We address this problem by proposing that we select the most soluble maximal relaxation as our explanation.

Approaches have been proposed that attempt to be more "helpful" by presenting users with partial consistent solutions [13], or advise on how to relax constraints in order to achieve consistency [11, 12]. Our approach is complementary to these by providing a basis for selecting from amongst the set of alternative explanations.

Recent work has focused on finding minimal unsatisfiable subproblems in temporal problems [9], satisfiability [7, 10] and type error debugging [2]. These techniques find all minimal unsatisfiable sets of constraints, which can be exponential in the number of constraints. Our work can be seen as a generalisation of these algorithms to the case where consistency is determined using an automaton.

7 Conclusions

We have considered the problem of generating maximal relaxations that are consistent with the largest number of solutions to the original problem. We studied this in the context of product configuration, where the constraint model of the configuration problem has been compiled into an automaton.

Two novel algorithms were presented. The first algorithm finds from amongst the longest relaxations to a set of inconsistent user constraints, the one that is consistent with the largest number of solutions; while the second finds from amongst the set of maximal relaxations, the one that is consistent with the largest number of solutions. Based on experiments on a large real-world automotive configuration problem we demonstrated the value of our approach.

In the future we intend to study how our algorithms generalise to more compact compiled representations of configuration problems, such as Decomposable negation normal form (DNNF) [4]. DNNF encodings can be exponentially more compact than an automaton, and since the time complexity of our algorithms depends on the size of the compiled form, there are obvious opportunities for our approach. We also plan to implement our approach in a real-world configurator, so that user-studies can be performed. We will also combine our approach with a preference-based approach to explanation [8].

Acknowledgements

This work was supported by Science Foundation Ireland (Grant No. 05/IN/I886).

References

1. Jérôme Amilhastre, Hélène Fargier, and Pierre Marguis. Consistency restoration and explanations in dynamic CSPs – application to configuration. *Artif. Intell.*, 135:199–234, 2002.
2. James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.
3. James Bowen. Using dependency records to generate design coordination advice in a constraint-based approach to concurrent engineering. *Computers in Industry*, 22(1):191–199, 1997.
4. Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)*, 17:229–264, 2002.
5. Eugene C. Freuder, Chavalit Likitvivatanavong, Manuela Moretti, Francesca Rossi, and Richard J. Wallace. Computing explanations and implications in preference-based configurators. In *Recent Advances in Constraints*, LNAI 2627, pages 76–92, 2003.
6. Gerhard Friedrich. Elimination of spurious explanations. In *Proceedings of ECAI*, pages 813–817, 2004.
7. Eric Gregoire, Bertrand Mazure, and Cedric Piette. Boosting a complete technique to find mss and mus thanks for a local search oracle. In *Proceedings of IJCAI*, pages 2300–2305, 2007.
8. Ulrich Junker. QuickXplain: preferred explanations and relaxations for over-constrained problems. In *Proceedings of AAAI*, pages 167–172, 2004.
9. Mark H. Liffiton, Michael D. Moffitt, Martha E. Pollack, and Karem A. Sakallah. Identifying conflicts in overconstrained temporal problems. In *Proceedings of IJCAI*, pages 205–211, 2005.
10. Mark H. Liffiton and Karem A. Sakallah. On finding all minimally unsatisfiable subformulas. In *Proceedings of SAT*, pages 173–186, 2005.
11. Barry O'Callaghan, Barry O'Sullivan, and Eugene C. Freuder. Generating corrective explanations for interactive constraint satisfaction. In *Proceedings of CP*, pages 445–459, 2005.
12. Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu. Representative explanations for over-constrained problems. In *AAAI*, pages 323–328, 2007.
13. Pearl Pu, Boi Faltings, and Marc Torrens. Effective interaction principles for online product search environments. In *Web Intelligence*, pages 724–727, 2004.
14. Mohammed H. Sqalli and Eugene C. Freuder. Inference-based constraint satisfaction supports explanation. In *Proceedings of AAAI*, pages 318–325, 1996.
15. Nageshwara Rao Vempaty. Solving constraint satisfaction problems using finite state automata. In *AAAI*, pages 453–458, 1992.

Using Support Vector Machines and Acoustic Noise Signal for Degradation Analysis of Rotating Machinery*

Patricia Scanlon and Susan Bergin

Bell Labs Ireland - Alcatel Lucent
Blanchardstown Industrial Park, Dublin 15, Ireland
{scanlon,bergin}@alcatel-lucent.com

Abstract. An automated approach to degradation analysis is proposed that uses a rotating machines acoustic signal to determine Remaining Useful Life (RUL). High resolution spectral features are extracted from the acoustic data collected over the entire lifetime of the machine. An information theoretic feature subset selection method is applied to remove redundant and irrelevant features. Using subsets of the feature space, multi-class linear and Radial Basis Function (RBF) Support Vector Machine (SVM) classifiers are developed and a comparison of their performance is provided. Performance of all classifiers is found to be very high, 85% to 98%, with RBF SVMs outperforming linear SVMs when a smaller number of features are used. As larger numbers of features are used for classification, the problem space becomes more linearly separable and the linear SVMs are shown to have comparable performance. A detailed analysis of the misclassifications is provided and an approach to better understand and interpret costly misclassifications is discussed.

Key words: Support Vector Machines, Degradation Analysis, Acoustic Signal Processing, Feature Selection

1 Introduction

Automated monitoring of machines typically involves the detection and diagnosis of defects. However in industry, there is an increasing demand for machine reliability and optimal management of spare parts inventory to prevent machine downtime. This demand requires such machine monitoring systems to also predict the Remaining Useful Life (RUL) of the machine in order to schedule materials, logistics and maintenance.
Machine monitoring systems reported in the literature are predominantly focused on defects developing in the components of Rolling Element Bearings (REBs). Detecting and diagnosing defects in REBs by manual visual inspection of the vibration or acoustic measurements has been reported in the literature [1–3]. However, such manual approaches are inefficient, require expert training and are

* This work was supported by IDA Ireland.

subjective.
Automated approaches to machine monitoring have been the focus of much research in recent times. These approaches can be broadly divided into detection, diagnosis and degradation analysis. Defect detection is a dual case problem where the classifier determines whether a defect exists or not. Defect diagnosis is a multi-class problem which attempts to classify which type of defect exists [4, 5]. Degradation analysis is also approached as a multi-class problem where developing defects are classified to several 'wear states' in order to determine the RUL of the machine [6].
Artificial neural networks (ANNs) have been applied in automated detection and diagnosis of machine faults [5] and degradation analysis for determining a bearings RUL [6]. In [4] Euclidean, Mahalanobis, and Bayesian distance classifiers, learning vector quantization (LVQ) classifier and the fuzzy gradient classifier are used for classification of various defects in washing machines vibration signals. Recent work in machine fault detection has employed the use of SVMs with RBF kernel [5, 7]. In this work the task of determining the RUL of rotating machinery is approached using SVM classifiers using both linear and RBF kernels.
Some early studies used sound pressure signals to explain the mechanism of vibration and noise generation in bearings [8, 9]. Several studies have shown that sound intensity, sound pressure and vibrational data measurements provide enough information to manually differentiate between 'good' and 'bad' bearings using spectral analysis [1] and statistical analysis [2]. An indepth review is given in Tandon [9] on the application of vibration measurements to both manual and automated machine monitoring systems as well as studies which examine acoustic measurements for manual defect detection. While vibrational signal measurements have been used in automated approaches to machine monitoring, to the best of our knowledge, no automated approach to defect detection, diagnosis or degradation analysis using acoustic measurements has been reported in the literature. Note that while vibrational analysis requires contact with the machine being monitored, acoustic analysis is advantageous as it allows for remote monitoring.
Various signal analysis techniques have been used in condition monitoring on both acoustic and vibrational data. These can be broadly classified into time domain techniques such as root-mean-square, crest factor, and statistical parameters such as mean, variance, skewness and kurtosis [9, 2] and frequency domain techniques such as cepstral, Short Time Fourier Transform (STFT), Wavelet Transform (WT) and envelope detection [10, 3, 9].
The studies on machine monitoring reported in the literature describe interesting approaches to this problem. However, the focus of the research described in this paper is an automated approach to machine monitoring using acoustic noise measurements to determine the RUL, which has not been addressed before in the literature. This study focuses on two key aspects of this novel system: first the selection of relevant spectral features extracted from the machines acoustic noise signal and second, the application of SVM classifiers with linear and RBF kernels to the multi-class classification problem of determining the RUL.

As spectral feature extraction results in irrelevant, noisy parts of the spectrum being included in the feature vector, an information theoretic approach to feature subset selection is employed to remove noisy features and select a compact and relevant feature set for classification. This approach does not require a-priori information regarding the spectral location of potential defects. This feature subset is used as input to multi-class linear and RBF SVM classifiers and a comparison of their performance is provided.
The following section describes the characteristics of bearing noise and how degradation might affect the noise is described as well as the feature extraction and feature subset selection. Section 3 describes the SVM classifiers used in the experiments. Section 4 describes the experimental setup and implementation and Section 5 describes and discusses the results of the experiments.

2 Acoustic Signal Processing

Rotating machinery is comprised of several moving parts including one or more Rolling Element Bearings (REB). Each element of the REB has a characteristic rotational frequency, when a defect develops on a particular element this frequency may get excited and energy increases at this spectral location. In normal operating conditions it is generally not known a-priori which defects will develop, whether multiple defects will develop and the severity of manufacturing variances. As a result the theoretical and actual defect frequencies will vary. Therefore, the development of a feature selection technique that does not require a-priori knowledge as to the spectral location of defect frequencies, such as the one described in the following section, is highly suitable to the task of Rotating Machine Monitoring.

2.1 Feature extraction

Spectral analysis decomposes the acoustic noise signal into frequencies so the influence of individual mechanical components can be ascertained as each type of fault has it's own characteristic spectral signature. A compact and relevant representation of the acoustic data is required as input to the classifier in order to determine the RUL of the machine. Conventional spectral features are first extracted from the data. The acoustic data is split into windowed non-overlapping time frames and spectral components are extracted using the Fast Fourier Transform (FFT) which results in an estimate of the short-term, time-localized frequency content of the acoustic signal. The spectral features are averaged over 20ms of acoustic data. The duration of the window has a pronounced effect on the nature of the features. A short duration window will result in good time resolution but yet degraded frequency resolution and the converse is also true Given the steady state nature of the signal acquired from the rotating machinery being monitored, time resolution is less significant than frequency resolution for degradation analysis.

As defects develop, the amplitude at the location of the defect frequency increases. If the frequency resolution is too coarse, this may cause relevant defect frequency information to be hidden during the early stages of defect development. However increasing frequency resolution increases dimensionality of the feature vector. While the relevant information may be uncovered, a considerable amount of irrelevant features will be included in the feature vector. Therefore, in order to effectively detect defects and yet eliminate irrelevant noisy features, feature subset selection is required to extract a limited number of relevant features for degradation analysis in order to determine the RUL.

2.2 Feature Subset Selection

Mutual Information (MI) can be used as a basis for selecting particular features to optimize the choice of inputs to a classifier [11]. MI can be defined as the reduction in entropy of one variable once another is known. The entropy of a random variable is a measure of its unpredictability. Specifically, if a variable X can take on one of a set of discrete values $\{x_i\}$ with a probability $Pr(X = x_i)$ then its entropy is given by:

$$H(X) = -\sum_{x \in \{x_i\}} Pr(X = x) \log Pr(X = x) \,, \tag{1}$$

If a second random variable C is observed, knowing its value will in general alter the distribution of possible values for X to a conditional distribution, $p(x|C = c)$. Because knowing the value of C can, on average, only reduce our uncertainty about X, the conditional entropy $H(X|C)$ is always less than or equal to the unconditional entropy $H(X)$. The difference between them is a measure of how much knowing C reduces our uncertainty about X, and is known as the Mutual Information (MI) between C and X, $I(X;C) = H(X) - H(X|C) = H(C) - H(C|X)$.
Further, $0 \leq I(X;C) \leq \min\{H(X), H(C)\}$, and $I(X;C) = 0$, if and only if X and C are independent. To obtain estimates of the MI values, the histogram approach was used to approximate the density functions required to estimate $p(X|C)$ and $p(X)$. The number of bins used was determined using Doane's rule [12], $K = \log_2 n + 1 + \log_2(1 + \widehat{k}\sqrt{n/6})$. In this rule, $\widehat{k}$ is the estimate of the kurtosis of the magnitudes of the spectral components (i.e., of random variable X), and n is the total number of training samples.
To compute an MI estimate for each candidate feature the class labels of the data are required. However, in the task of Machine Monitoring, data is acquired over the lifetime of the machine. The data can be divided into gradually increasing stages of wear or classes C. However, the location in time of the boundaries between these classes is unknown. Only the condition of the machine at the beginning and end of the lifetime are certain, in between is a progression of failure. To remove any a-priori decisions regarding class boundaries and also take advantage of the fact that there is a chronological order to the collected data samples, classes C are defined as the set of short fixed length overlapping

time frames over the lifetime of the machine. This novel approach computes the average local entropy from the entropy of each overlapping time frame over the lifetime and subtracted from the global entropy (over the entire lifetime) to obtain the MI estimate for each feature. The most relevant features are selected based on the highest MI criterion, and MI is computed for each spectral feature in isolation. While increasing dimensionality is desirable to uncover the relevant defect frequencies for monitoring, too high a resolution will result in features that are highly correlated with their immediate spectral neighbours. This can lead to redundant features in the final feature vector used as input to the classifier.

3 Classification

Support Vector Machines (SVMs) are a relatively recent set of supervised machine learning algorithms that have been shown to have either equivalent or significantly better generalization performance than other competing methods on a wide range of classification problems [13]. They can be used to classify linearly separable data using the original input space or non-linearly separable data by mapping to a higher dimensional feature space in which a linear separator can be found.
In a typical binary classification problem composed of a training dataset $\{(\mathbf{x_1},y_1), (\mathbf{x_2},y_2), ..., (\mathbf{x_m},y_m)\}$ where $\mathbf{x_i} \in \Re_\mathbf{d}$ and $y_i \in \{\pm 1\}$, SVMs seek a solution to the following Lagrangian optimization function:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2}$$

subject to the following constraints

$$C \geq \alpha_i \geq 0 \quad \forall i \qquad and \qquad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{3}$$

C is an optional parameter that controls the trade off between allowing training errors and forcing rigid margins. That is, it represents a soft margin that allows some misclassifications which can be beneficial in noisy datasets. Where a soft margin is not allowed, the constraint is simply $\alpha_i \geq 0$. K represents the kernel function and numerous choices exist, including:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \cdot \mathbf{x}_j + 1)$
- Polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$
- Radial Basis Function: $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$, $\gamma > 0$.

Once an optimal solution is found, the decision function for a new point $\mathbf{z}$ is given by

$$f(\mathbf{z}) = sign\left(\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) + b\right). \tag{4}$$

z is a training example, b is the bias and non-zero α_i values represent support vectors, the points that lie closest to the hyperplane.
In this study two SVM kernels are implemented - a linear SVM and an RBF SVM. Linear SVMs have very fast training times and do not require any parameter tuning (except C when soft margins are used). Moreover, if the samples can be correctly classified using a linear decision boundary the computational complexity associated with non-linear kernels can be avoided. RBF SVMs are currently the most popular choice of non-linear SVM and thus are an appropriate algorithm for a first experiment in this problem space [13], [14].
A number of approaches have been proposed to extend SVMs to handle multi-class classification problems, for example, one-against-all, one-against-one and directed acyclic graph SVM (DAGSVM). 'One-against-one' [15] is implemented in this study as it has been shown to have comparable if not better generalized accuracy than alternative techniques and requires considerably less training time [16], [17]. The method consists of constructing an SVM for each pair of classes. Thus for a problem with n classes $n(n-1)/2$ SVMs are trained to distinguish between the samples of one class form the samples of another class. For an unknown pattern, each SVM votes for one class and the class with the highest number of votes is chosen.

4 Experimental Setup and Implementation

Acoustic data was collected from a rotating machine running at approximately 2200RPM in high heat conditions to accelerate failure over a period of approximately 6 months. The final failure was due to complete bearing seizure. Acoustic data was acquired at a sampling rate of 50,000 samples/second over the lifetime of the machine. Short-term log-spectral features were extracted from the acoustic signal using 1024 point FFT to provide sufficiently high frequency resolution to uncover the relevant information required for effective automated monitoring that would otherwise be hidden using a coarser resolution. This resultant spectral features span the entire spectrum from 0-25kHz. Information pertinent to machine monitoring occurs at localised spectral locations across the spectrum. The rest of the spectrum contains background noise that is irrelevant to machine monitoring. To remove such noisy components the MI approach to feature subset selection is employed. The average MI over the entire 513 extracted spectral features is approximately 0.5 with a standard deviation of 0.25, maximum MI is 1.6. Therefore, only the 200 features with a MI value above the mean were considered for use in classification. To investigate smaller sets of features that could be used to predict RUL with high accuracy, subsets of 25, 50, 100 and 150 features were also examined.

In order to perform degradation analysis a classification approach is proposed that determines what state of degradation, or 'wear states', the machine is currently in. In this paper 10 wear states are used to predict the RUL. The degradation of the machine progresses through several stages of physical wear. As the exact location in time where such degradation events occur is difficult to

ascertain, it is assumed that the machines degradation is a progression of wear and the data is divided into 10 equal segments for labeling. Each segment or wear state represents a different time interval over the lifetime of the machine from 1 to 10, where 1 is new and 10 is approaching failure.

The procedure taken to implement the linear SVMs was as follows. First the attributes are scaled to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges. A fixed penalty parameter (C) of 1.0 was used. This is important, if high performance results can be achieved using this default value, then lengthy parameter tuning can be avoided. Generalization accuracy was determined using 10-fold stratified cross validation. In this procedure, data is randomly split into 10 parts, with each part representing the same proportion of each class or wear state. Each part is held out in turn and the learning scheme is trained on the remaining nine parts. The error rate is calculated on the holdout (test) set. The procedure is executed 10 times on different training sets and the results are averaged over all of the testing datasets. Although this approach is more computationally intensive than the commonly used 'hold out' method, all examples in the dataset are used for training and testing and thus confidence on the generalisability of the results is increased. In addition the stratification process improves the representativeness of each fold as the process seeks to represent the same proportion of each class in a fold as is in the original full dataset.
The implementation process for the RBF kernel was more involved as two parameters (γ, the kernel parameter and C) are required. First, the attributes are scaled. Then a grid search using 10-fold cross validation was performed to find the best γ and C parameters. The identified values were then used to train 66% of the training set and the generalization accuracy rate was determined using the remaining test instances.

5 Results and Discussion

The results from the two SVM procedures outlined in the previous section are provided in Table 1. From Table 1 it can be seen that when fewer features ($n = 25, n = 50, n = 100$) are used the RBF SVM significantly outperforms the linear SVM. However, the difference is far less pronounced where more features are used ($n = 150, n = 200$). It appears that the problem becomes more linearly separable as more features are included. Furthermore, the training time of the linear kernel is significantly less than that of the RBF kernel, for example on the 200 feature set problem, the RBF takes approximately 25 times longer to run on the same machine where no other processes were active. Although, both provide very high performance, this paper promotes the use of a linear SVM using 200 features for several reasons. First, it is far less computationally intensive than the RBF SVM and achieves comparable results. Second, as outlined earlier, it has a considerably faster training time. This is an important consideration from a company perspective where delays in deploying a monitoring component in the field potentially results in increased costs and lost revenue. Third, it is

intuitively easier to understand and interpret the problem space and solution. However, additional studies evaluating alternate classifiers in this problem space are warranted. All further results discussed in this paper are based on the 200-feature linear SVM.

Further analysis of the results is valuable to augment preventative maintenance scheduling and machine replacement. In terms of misclassification, predicting something is going to fail earlier than it truly will, results in, at worst, a waste of resources, for example preventive maintenance or machine replacement happens earlier than necessary. The cost incurred is a function of how early this maintenance or replacement takes place, for example, replacing a machine one time-interval before it would have failed is much more cost-efficient than replacing the machine six time-intervals before it might have failed. On the other hand, predicting something is likely to fail later than it actually does could be far more serious, causing, for example, machine down-time and customer dissatisfaction. To this end, an analysis of how 'inaccurate' the misclassifications are is a worthy addition to this study. As illustrated in Table 2, 47.6% of failures are predicted as happening earlier (1 time-interval earlier) than they truly do and as such are not considerably costly. Furthermore, the 2.4% predicted to have failed two time-intervals earlier than they truly would is arguably insignificant given how small an error this is. Thus, the real cost of misclassification is the 50.8% that are predicted as happening later than when they actually do and further analysis is required. Although SVMs typically only output a target label for each input,

Table 1. Comparison of linear and RBF kernel Support Vector Machine classifiers using subsets of the spectral, features based on the Mutual Information criteria

no. of features	**Linear SVM (%)**	**RBF Kernel (%)**
25	84.86	92.37
50	91.69	96.39
100	94.68	97.48
150	96.06	97.52
200	96.29	97.93

Table 2. Misclassification Analysis

Description	**misclassified (#)**	misclassified (%)
failed 1 time-interval later than predicted	179	47.6%
failed 2 time-interval later than predicted	6	1.6%
failed 3+ time-intervals later than predicted	0	0%
failed 1 time-intervals earlier than predicted	179	47.6%
failed 2 time-intervals earlier than predicted	9	2.4%
failed 3+ time-intervals earlier than predicted	3	0.8%
Total % of samples misclassified	376	(3.7%)

an extension to the algorithm is possible to generate probability estimates for each sample. The estimates are based on the distance each test point is from the separating hyperplane, the further the point is from the hyperplane, the higher the probability it belongs in the class [18]. Analysis of the probabilities reveals that for 94% of the misclassified instances the actual class had the next highest probability to the predicted class. This is very important as it provides a measure of confidence for the predicted RUL and allows preventative maintenance and replacement to be scheduled in a more knowledgeable fashion.
Further analysis of results in Table 2 were carried out to determine exactly where in time misclassifications were made. Most of the misclassifications appeared around the middle and late classes or time-intervals and this suggests that equal time segmentation is not the most suitable technique for class boundaries in these regions. A more sophisticated separation of the class boundaries, for example using a clustering technique, could further improve this misclassification error.

6 Conclusion

Commonly employed machine monitoring techniques, such as measuring changes in speed and current are only useful in indicating when failure is imminent. In order to the RUL of the machine to increase reliability and decrease machine downtime more sophisticated measurements such as vibrations and acoustic noise can be used. Employing the use of acoustic noise measurements as opposed to the commonly used vibrational signal allows for remote, non-contact, monitoring of the machine. Automated machine monitoring using the acoustic noise signal to determine RUL, to the best of our knowledge, has not been attempted before in the literature.
The results of the experiments described in this paper have indicated that there exists sufficient information in the acoustic noise signal of rotating machines in order to effectively determine the RUL. A novel approach to feature subset selection on a high dimensionality spectral feature vector was proposed in order to remove noisy spectral components before classification using MI without class boundary labels. In addition this approach does not require spectral locations of the defect frequencies to be defined a-priori. In addition, the detailed misclassification analysis provides valuable knowledge and a measure of confidence in the predictions that can be used to further optimize preventative maintenance scheduling and machine replacement. Finally, it is hypothesised that further improvements in performance can be achieved, for example, by using an automated clustering approach to determine the wear state boundaries for classification.

References

1. N. Tandon and B. Nakra, "The application of the sound intensity technique to defect detection in rolling element bearings," *Applied Acoustics*, vol. 29:3, pp. 207–217, 1990.

2. R. Heng and M. Nor, "Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition," *Applied Acoustics*, vol. 53, pp. 211–226(16), 1998.
3. D. Shi, W. Wang, and L. Qu, "Defect detection for bearings using envelope spectra of wavelet transform," *Journal of Vibration and Acoustics*, vol. 126:4, pp. 567–573, 2004.
4. S. Goumas, M. Zervakis, and G. Stavrakakis, "Classification of washing machines vibration signals using discrete wavelet analysis for feature extraction," *IEEE Trans. on Instrumentation and Measurement*, vol. 51:3, pp. 497–508, 2002.
5. B. Samanta, K. Al-Balushi, and S. Al-Araimi, "Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection," *Engineering Applications of Artificial Intelligence*, vol. 16:7, pp. 657–665(9), 2003.
6. H. Lao and S. Zein-Sabatto, "Analysis of vibration signal's time-frequency patterns for prediction of bearing's remaining useful life," *Proc. of the 33rd Southeastern Sym. on System Theory*, vol. 1, pp. 25–29, 2001.
7. A. Rojas and A. Nandi, "Detection and classification of rolling-element bearing faults using support vector machines," *IEEE Workshop on Machine Learning for Signal Processing*, pp. 153 – 158, 2005.
8. V. Jayaram and F. Jarchow, "Experimental studies on ball bearing noise," *Wear*, vol. 46, pp. 321–326, 1978.
9. N. Tandon and A. Choudhury, "A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings," *Tribology International*, vol. 32, pp. 469–480, 1999.
10. P. McFadden and J. Smith, "Vibration monitoring of rolling element bearings by the high frequency resonance technique a review," *Tribology International*, vol. 17:1, pp. 1–18, 1984.
11. P. Scanlon, D. Ellis, and R. Reilly, "Using broad phonetic group experts for improved speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 15:3, pp. 803–812, 2007.
12. D. Doane, "Aesthetic frequency classifications," *The American Statistician*, vol. 30, pp. 181–183, 1985.
13. C. Burges, "A tutorial on support vector machines for pattern recognition," *Knowledge Discovery and Data Mining*, vol. 2, no. 2, pp. 121–167, 1998.
14. B. ScholKopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
15. S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," *Neurocomputing: Algorithms, Architectures and Applications*, 1990.
16. C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.
17. J. Milgram, M. Cheriet, and R. Sabourin, "'one against one' or 'one against all': Which one is better for handwriting recognition with svms?" *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.
18. J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," *Advances in Large Margin Classifiers*, pp. 61–74, 2000.

Constructive vs Perturbative Local Search for General Integer Linear Programming

Stefania Verachi and Steven Prestwich

Cork Constraint Computation Centre
Department of Computer Science, University College, Cork, Ireland
{s.verachi,s.prestwich}@cs.ucc.ie

Abstract. Most local search algorithms are "perturbative", incrementally moving from a search state to a neighbouring state while performing noisy hill-climbing. An alternative form of local search is "constructive", repeatedly building partial solutions using greedy or other heuristics. Both forms have been combined with constraint propagation, and they can be hybridised with each other by perturbing partial solutions. We design a new hybrid constructive local search algorithm for general integer linear programs, combining techniques from constraint programming, boolean satisfiability, numerical optimisation and scheduling. On a hard design problem it scales better to large instances than both a perturbative algorithm and a Benders decomposition algorithm.

1 Introduction

A currently active area of research is the hybridisation of constraint programming (CP) techniques with those of artificial intelligence (AI) and operations research. By combining their different strengths we may solve problems that are considered otherwise practically unsolvable. One branch of this research aims to combine the space-pruning ability of constraint propagation with the scalability of local search. For example replacing systematic backtracking by a non-systematic form called Incomplete Dynamic Backtracking (IDB) boosts scalability to equal that of local search [16], and in fact is claimed to be local search in a different search space. IDB has been implemented mainly using variations on forward checking (FC). Another example is Decision-Repair [10], designed for constraint satisfaction problems (CSPs) and applied to open shop problems. It uses learning and has heuristics such as clause weighting, a TABU list and greedy hill climbing.

However, there are drawbacks with these hybrids. Like most local search algorithms, they are *perturbative*: they incrementally move from one search state to another, while combining hill-climbing with noise and other heuristics. Unfortunately, incrementally maintaining consistency can be quite complex and require expensive data structures, especially for higher forms of consistency. An alternative *constructive* form of local search has received relatively little attention [8] but is a promising approach to hybridisation. (For the rest of this paper we shall abbreviate *constructive local search* to *CoLS* and *perturbative local search* to *PeLS*.) CoLS algorithms behave like backtracking algorithms until the first

backtrack would normally occur, at which point they restart the search. This repeated construction of partial solutions may be combined with a variety of techniques including greedy algorithms, constraint propagation and relaxation. An advantage of CoLS over PeLS is that it is much easier to hybridise with constraint propagation techniques, because exactly the same implementation techniques can be used. A disadvantage is that frequent restarting incurs significant overheads, but despite this CoLS has been shown to pay off on several problems. For example Squeaky Wheel Optimization (SWO) beat TABU search on a set of scheduling problems, and it beat TABU and three specialised local search algorithms on graph colouring problems [9]; a version of SWO also beat a version of IDB on some generalised graph colouring instances though IDB won on other instances [18]; and UnitWalk [7] beat several perturbative algorithms on industrial benchmarks in SAT solver competitions.

Besides the obvious algorithmic differences, there is another important difference between local search and CP algorithms, leading to a design decision regarding hybrids. With most forms of propagation we must represent entire variable domains in order to keep track of pruned values. In contrast, most local search algorithms do not maintain domains but simply remember the current value of each variable. Their memory requirements are therefore independent of the domain sizes, which is a significant advantage for problems with large domains. In order to combine the power of constraint propagation with the low memory requirements of local search, we use only a restricted form of consistency: *bounds consistency* (BC). This restriction has the advantage that for each integer variable we maintain only an upper and a lower bound denoting the values currently in the domain, making the memory requirement independent of domain size. BC also has the advantage of cheap propagation algorithms [24].

Another design decision is the choice of constraint language, and we choose general Integer Linear Programming (ILP). Many theoretical and real-world combinatorial problems have elegant ILP models, so local search algorithms for ILP immediately have potential applications. ILP also fits very well with the choice of BC: though BC is generally a rather weak form of consistency, in the special case of *linear inequalities* (and for a more general class of *monotonic constraints*) it is equivalent to *generalised arc* (or *hyper-arc*) *consistency* (GAC) [28]. GAC is a strong form of consistency that is much used in CP, so for this restricted class of problems we can design a hybrid local search algorithm that maintains a high level of consistency. Any equality constraints in an ILP can be reduced to inequalities: this weakens the propagation but it is impractical to achieve BC on equalities, which is an NP-hard problem [28].

Thus the proposed algorithm has a combination of features that should make it a useful tool for a range of large structured problems: low memory requirements, powerful but cheap constraint propagation, local search scalability, no novel implementation techniques, and a well-known and reasonably expressive modelling language. The paper is structured as follows: Section 2 describes our algorithm, Section 3 reports the results of experiments, Section 4 describes related work, and Section 5 concludes the paper and discusses future work.

2 The Algorithm

Our algorithm, which we call BOLOS (Bounds-Oriented LOcal Search), is similar in structure to Adaptive Iterated Construction Search [8]. Its original inspiration was the UnitWalk algorithm for SAT [7] but it also takes ideas from other algorithms including SWO [9]. An outline of the core of BOLOS, which finds feasible solutions to a problem π, is shown in Fig. 1 (we discuss solution quality below).

```
procedure BOLOS-feasible(π):
    s ← ∅, w ← 0
    make π locally consistent
    if inconsistent then return "no solution"
    while not feasible(π, s)
        s ← construct(π, w, s)
        s ← perturb(π, s)
        w ← prioritise(π, s, w)
    return s
```

Fig. 1. BOLOS core for finding feasible solutions

The initial partial solution is empty. With each problem variable is associated a weight, and the weights are initialised to zero. Constraint propagation is applied before search begins to enforce local consistency, in this case Bounds Consistency (BC). If BC establishes unsolvability then the algorithm terminates. Each iteration has three phases: firstly a partial candidate solution s is constructed, guided by the weights and the previous partial solution; secondly perturbative local search is applied to improve s; thirdly the weights w are adjusted. Termination occurs when all constraints are satisfied. Section 2.1 describes partial solution construction, Section 2.2 perturbative local search, Section 2.3 variable prioritisation by weight adjustment, and Section 2.4 the extension to optimisation problems.

2.1 Construction

A partial solution is constructed by selecting a variable, assigning a value to it, propagating the result to the other variable domains, and repeating until either no unassigned variables remain or *domain wipeout* occurs (constraint propagation reduces the domain of at least one variable to the empty set, implying that the partial solution cannot be extended to a full solution). If domain wipeout occurs then the variable whose assignment led to the wipeout, and the variables whose domains were wiped out, are labelled as *troublemakers*, and this information is used during prioritisation (see Section 2.3). Apart from the troublemaker heuristic, this is identical to the behaviour of a standard constraint solver prior to its first backtrack.

Variables are selected first by smallest domain size (a common CP heuristic), ties are broken by *prioritisation* using the dynamic weights (see Section 2.3), and further ties are broken randomly. The value selected for each variable is, where possible, the same value used in the previous iteration after perturbation (random values are assumed before the first iteration). If this value is not currently in the variable domain then another value is chosen as follows: if the current lower bound is greater than the previous value then the lower bound is used; otherwise, if the current upper bound is smaller than the previous value then the upper bound is used.

The form of constraint propagation used is BC, which can be applied to many types of constraint problem. BC has low computational overhead, since each domain is represented only by its lower and upper bound. There is more than one BC algorithm even for the case of linear inequalities on integer variables [6, 28], and it is also possible to perform a weaker form of consistency called Bounds Propagation (BP) which is a form of forward checking using bounds reasoning. The difference in the implementation between BC and BP is that BC requires a queue for arc revision, to maintain the violated constraints that must be revisited whenever a bounds update will be made. BP only propagates to future (currently unassigned) variables and does not require a queue, so it sacrifices some propagation but reduces runtime overhead. It is known that stronger forms of consistency pay off in some cases [22] but not others [11], and we have found that BC is sometimes worthwhile and sometimes not; in this paper we actually use BP.

2.2 Perturbation

To improve the quality of a partial solution we apply a simple perturbative local search algorithm to try to reduce the number of constraint violations. Perturbative local moves are usually cheap compared to constructive iterations so this adds relatively little runtime overhead. We also execute the perturbation phase once at the start of the algorithm, to provide a reasonable initial state.

In local search for CP and SAT, in which we try to remove violations, perturbative moves are usually applied to complete variable assignments. In BOLOS we only have partial assignments, but we overcome this problem by constructing a total assignment consisting of the current assignments plus the most recent assignments of the unassigned variables. The perturbation works by randomly selecting a constraint that is violated under this total assignment, randomly selecting a variable in the constraint, and reassigning that variable so that the constraint is just satisfied. If no domain value satisfies the constraint then either the upper or lower bound (whichever is best) is used to reduce the degree of violation. The resulting values are used as the default values in the next iteration. This Gauss-Seidel-like technique was taken from numerical optimisation (see standard textbooks in this area). A design decision concerns how long to run the perturbation phase. We apply it in a limited way, visiting each constraint at most once, and ignoring any new violations generated during perturbation.

We also apply a form of *noise* by randomising the assignment of a randomly-chosen variable — in this paper we do this every 10 iterations (UnitWalk only does this on detecting that no variable was reassigned during an iteration). We also apply an extreme form of perturbation: *random restarts*. New random values for all variables are generated after k iterations, where k is a runtime parameter tuned by the user for each problem. In this paper we set $k = 2000$.

2.3 Prioritisation

As mentioned in Section 2.1 some variables are labelled as "troublemakers" during the construction phase, and we hope that by focusing attention on these variables we can more quickly move toward a solution. This kind of dynamic prioritization is used in several AI algorithms, including some scheduling algorithms and the VSIDS branching rule in some complete SAT algorithms. Each variable has an associated weight that changes dynamically during search. Large weights denote more troublesome variables, and during construction these variables are preferred.

We use a simple scheme to update the weights: they are all initially 0, and on defining a new set of troublemakers their weights are assigned to n (the number of variables in the problem), and all other weights are decremented by 1 (unless they are already 0). However, for the problem described in this paper, we found that a variant in which the weights are not decremented worked better. This has the effect of prioritising variables during the first few iterations, but not later when all variables have been assigned top priority. For most of the search the variables are therefore ordered using only the smallest-domain branching rule.

2.4 Optimisation

If (as is usual) the ILP has a linear objective function then BOLOS as described above must be extended. Assuming without loss of generality that the function is to be minimised, we use a simple approach: start with a very high (or infinite) upper bound on the objective, expressed as another linear constraint; solve the feasibility problem using BOLOS; then tighten the constraint and repeat, until reaching either a known optimum cost, or timing out, or the initial application of BC detects infeasibility. Variable assignments and priorities are retained between iterations so that the search for a new solution starts in the neighbourhood of the last solution. We also use a form of intensification described in [2, 8]: every f iterations (for some user-defined integer f) we return to the last feasible solution found. In this paper we set $f = 130$.

3 Application to Stochastic Template Design

The original template design problem was first described by Proll & Smith [20] who observed it at a local colour printing firm producing a variety of products from thin board, including cartons for human and animal food and magazine

inserts. The problem is described as follows. Given a set of variations of a design, with a common shape and size and such that the number of required *pressings* of each variation is known. The problem is to design a set of templates, with a common capacity to which each must be filled, by assigning one or more instances of a variation to each template. A design should be chosen that minimises the total number of *runs* of the templates required to satisfy the number of pressings required for each variation. As an example, the variations might be for cartons for different flavours of cat food, such as fish or chicken, where ten thousand fish cartons and twenty thousand chicken cartons need to be printed. The problem would then be to design a set of templates by assigning a number of fish and/or chicken designs to each template such that a minimal number of runs of the templates is required to print all thirty thousand cartons. Proll & Smith [20] address this problem by fixing the number of templates and minimising the total number of pressings.

The problem was extended to demand uncertainty via scenarios by Tarim & Miguel [25]. They used a probabilistic model in which demands are random variables, and added *scrap cost* and *shortage cost*. They proposed a certainty-equivalent, non-linear model for this generalised problem, in which they minimise the expected total cost. They solved the problem using a Benders decomposition algorithm (with single cuts) that was shown to be superior to two other versions (with complete and multiple cuts) and to a stochastic constraint programming method. Prestwich, Tarim & Hnich [19] later linearised this model and solved it using a local search algorithm (VWILP) for ILP models, obtaining better results as the allowed number of templates increased.

We applied BOLOS to the same benchmark set as [19, 25] using the same ILP model as [19] (details omitted for space reasons) and under the same experimental conditions: for each problem instance the algorithm was run once with a cutoff time of 1 hour, and the cost of the best solution found in that time recorded along with the actual time taken to find it. The results from [19] used a 2 GHz Intel Centrino, 1 GB RAM machine, and we normalised our runtimes to that machine by comparing runtimes for our algorithm on that machine and ours (a 2.8 GHz Pentium (R) 4 with 512 RAM), which was almost exactly 3 times faster. We compare the complete Benders algorithm with VWILP and BOLOS. The results are shown in Tables 1 and 2 for two, three and four templates. Lowest costs in each case are shown in **bold**.

Clear patterns emerge over the 60 instances: for 2 templates the winner is Benders, for 3 templates VWILP, and for 4 templates BOLOS. As noted in [19], as the number of templates increases Benders finds poorer solutions within the cutoff time, though the optimal solution can only get better and the local search algorithms do find better (optimal or suboptimal) solutions. Comparing VWILP and BOLOS only: with 2 templates VWILP wins in 15 cases and BOLOS in 1 case; with 3 templates VWILP wins in 10 cases and BOLOS in 7 cases; with 4 templates VWILP wins in 5 cases and BOLOS in 14 cases. In summary, despite BOLOS's greater runtime overhead than VWILP, it scales better to more templates, and beats 5 other known algorithms applied to these benchmarks.

	Benders		VWILP		BOLOS			Benders		VWILP		BOLOS	
no	cost	time	cost	time	cost	time	no	cost	time	cost	time	cost	time
1	**285.00**	10	**285.00**	230	**285.00**	25	1	295.50	68	**285.00**	160	285.50	1914
2	**285.00**	8	**285.00**	990	**285.00**	1107	2	305.50	110	**285.00**	440	285.50	3150
3	**480.00**	62	**480.00**	4	562.50	303	3	309.00	3600	**307.50**	2100	**307.50**	681
4	**332.50**	41	**332.50**	6	412.50	195	4	462.00	110	**310.00**	99	332.50	423
5	**322.50**	190	**322.50**	196	332.50	513	5	465.00	440	310.00	690	**309.50**	2406
6	401.00	3400	**365.00**	140	**365.00**	114	6	481.00	180	**365.00**	310	**365.00**	378
7	**308.00**	670	315.00	570	327.50	180	7	805.00	1100	**307.50**	1600	314.00	1683
8	**310.00**	820	312.50	170	317.50	102	8	464.50	1100	312.50	230	**305.50**	2151
9	**308.00**	1800	310.00	1500	319.00	3582	9	471.00	1600	312.50	390	**306.00**	1578
10	326.00	2900	**310.00**	1680	316.00	216	10	775.00	1500	310.00	3100	**307.50**	1203
11	**333.00**	480	339.00	3500	342.00	1998	11	770.00	620	**333.00**	87	342.00	3042
12	**354.00**	1100	362.00	47	362.00	705	12	747.25	630	**361.75**	3300	362.25	348
13	**374.00**	190	379.50	2	384.50	696	13	557.00	290	372.00	2200	**370.75**	549
14	**393.50**	1300	396.00	420	399.50	783	14	522.25	420	**393.50**	1300	**393.50**	540
15	**407.00**	800	414.50	3000	423.75	795	15	531.25	410	**407.00**	570	408.00	3282
16	**432.50**	3600	433.25	44	474.00	1662	16	544.25	820	**428.25**	2900	447.00	498
17	**398.25**	540	400.75	220	435.75	267	17	526.50	300	**399.50**	2200	400.75	2283
18	**377.75**	290	380.75	740	392.50	390	18	512.75	220	**375.75**	3000	378.75	1737
19	**396.75**	1200	414.25	1700	413.25	2484	19	530.75	620	399.25	880	**398.00**	2868
20	**406.75**	400	409.25	110	425.50	2682	20	547.25	660	408.00	2900	**407.75**	1197

Table 1. Results with two (left) and three (right) templates

4 Related Work

A simple form of CoLS is Iterative Sampling [12] which simply restarts a backtrack-style algorithm at every dead end. Allowing a number of backtracks is known as *random restarting* [5]. This can be made complete and works well on many problems, but does not scale in a similar way to local search. UnitWalk [7] is a CoLS algorithm for SAT that uses unit propagation. It starts in the same way as a backtracker, selecting a variable, assigning a value to it, propagating where possible, and repeating, until a dead end is reached. It then restarts using a slightly different variable ordering. It has also been hybridised with perturbative local search to improve its performance on some benchmarks. SWO [9] is a CoLS algorithm that operates in two search spaces: a solution space and a prioritisation space. Both searches influence each other: each solution is analysed and used to change the prioritisation, which guides the search strategy used to find the next solution, found by restarting the search. Other examples of CoLS cited in [8] are a stochastic tree search algorithm [3] and Adaptive Probing [21], and it can also be viewed as a special case of Ant Colony Optimisation.

A few (mainly perturbative) local search algorithms for forms of integer program have been reported. General Purpose SIMulated ANnealing (GPSIMAN) [4] is an early algorithm for 0/1 problems using Simulated Annealing (SA). From a feasible assignment a variable is selected and its value flipped (reassigned from

	Benders		VWILP		BOLOS	
no	cost	time	cost	time	cost	time
1	295.50	860	**285.00**	470	285.50	279
2	545.00	82	**286.00**	3100	292.00	3162
3	2128.00	2100	310.00	35	**305.00**	2976
4	468.00	1100	307.50	2300	**305.00**	783
5	3478.00	0.0	310.00	210	**309.50**	423
6	481.00	93	**365.00**	430	**365.00**	1413
7	855.50	650	307.50	950	**305.50**	1308
8	2563.00	140	312.50	1700	**307.50**	1458
9	3478.00	0.0	312.50	320	**306.50**	909
10	3476.00	0.0	310.00	2200	**306.00**	2358
11	770.00	3300	335.00	1700	**333.50**	3255
12	1569.50	380	357.00	1900	**351.00**	819
13	557.00	1500	372.00	1200	**370.75**	3138
14	522.25	2200	**393.50**	1100	400.50	612
15	531.25	2200	407.00	2700	**406.75**	1713
16	2003.75	220	433.25	2400	**428.25**	786
17	526.50	1500	**397.00**	3200	402.00	2232
18	512.75	1100	379.50	1500	**377.50**	2535
19	530.75	3600	**399.25**	780	403.75	2397
20	1747.50	170	411.75	650	**410.50**	777

Table 2. Results with four templates

0 to 1 or vice-versa), then SA is used to restore feasibility. This approach was generalised to integer variables in [1]. Pedroso [15] describes an evolutionary algorithm for MIP that searches on the integer variables, then uses linear relaxation to fix the continuous variables. A related approach used a version of the Greedy Randomised Adaptive Search Procedure (GRASP) [14] for MIP. TABU search has also been applied to MIP, for example see [13]. *Filled function* methods are similar to perturbative local search and have also been used to solve ILPs [23]. Some SAT local search algorithms have been adapted to integer programs. WSAT(PB) [26] is a generalisation of the WalkSAT algorithm that applies to 0/1 problems, and WSAT(OIP) [27] is a further generalisation to integer variables. Saturn [17] is a hybrid of local search and constraint propagation for 0/1 problems, generalised from an earlier SAT algorithm. The closest algorithm to BOLOS is GRASP (see [14] and other works), a meta-heuristic that alternates constructive phases to find good solutions with local search phases to find locally optimum solutions.

5 Conclusion

We presented a hybrid constructive/perturbative local search algorithm for ILP and showed that, on a design problem, it scales up better than the best known

complete algorithm and a recently published perturbative algorithm. The algorithm contains a novel combination of techniques from constraint programming, SAT, numerical optimisation and scheduling.

We envisage several directions for future work. Firstly, like most local search algorithms, BOLOS has several runtime parameters and optional heuristics. In this paper we chose values for a given class of problems after some experimentation, but in future work we hope to freeze the choice of heuristics and eliminate at least some of the parameters. Secondly, we aim to improve BOLOS's heuristics, perhaps by extension to a population-based search such as an evolutionary or ant colony algorithm. Thirdly, we plan to make BOLOS applicable to a wider range of problems by extending its use of bounds consistency to other constraints besides linear inequalities, for example to more general monotonic constraints, or to global constraints such as `alldifferent`. Fourthly, we could extend BOLOS to MIP by applying the Simplex method to the continuous variables after fixing the integer variables, as in [14, 15].

Acknowledgements This material is based upon works supported by the Science Foundation Ireland under Grant Nos. 04/BR/CS0355 and 00/PI.1/C075.

References

1. D. Abramson, M. Randall. A Simulated Annealing Code for General Integer Linear Programs. *Annals of Operations Research* 86:3–24, 1999.
2. J. C. Beck. Solution-Guided, Multi-Point Constructive Search. *Journal of Artificial Intelligence Research*, 2007 (to appear).
3. J. L. Bresina. Heuristic-Biased Stochastic Sampling. *13th National Conference on Artificial Intelligence*, AAAI Press / The MIT Press, 1996, pp. 271–278.
4. D. Connolly. General Purpose Simulated Annealing. *Journal of the Operational Research Society* 43:495–505, 1992.
5. C. P. Gomes, B. Selman, K. McAloon, C. Tret. Randomization in Backtrack Search: Exploiting Heavy-Tailed Profiles for Solving Hard Scheduling Problems. *4th International Conference on Artificial Intelligence Planning Systems*, Pittsburgh, PA, 1998.
6. W. Harvey, J. Schimpf. Bounds Consistency Techniques for Long Linear Constraints. *Workshop on Techniques for Implementing Constraint Programming Systems*, 2002.
7. E. A. Hirsch, A. Kojevnikov. UnitWalk: A New SAT Solver That Uses Local Search Guided by Unit Clause Elimination. *Annals of Mathematics and Artificial Intelligence* 43(1–4):91–111, 2005.
8. H. H. Hoos, T. Stützle. Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco, CA, USA, 2004.
9. D. E. Joslin, D. P. Clements. Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research* 10:353–373, 1999.
10. N. Jussien and O. Lhomme. Local Search With Constraint Propagation and Conflict-Based Heuristics. *Artificial Intelligence* 139(1):21–45, 2002.
11. V. Kumar. Algorithms for Constraint Satisfaction Problems: a Survey. *AI Magazine* 13(1):32–44, 1992.

12. P. Langley. Systematic and Nonsystematic Search Strategies. *1st International Conference on Artificial Intelligence Planning Systems*, 1992.
13. A. Løkketangen, F. Glover. Tabu Search for Zero-One Mixed Integer Programming with Advanced Level Strategies and Learning. *International Journal of Operations and Quantitative Management* 1(2):89–108, 1995.
14. T. Neto, J. P. Pedroso. GRASP for Linear Integer Programming. Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, 2004, pp. 545–574.
15. J. P. Pedroso. An Evolutionary Solver for Linear Integer Programming. BSIS Technical Report 98-7, Riken Brain Science Institute, Wako-shi, Saitama, Japan, 1998.
16. S. D. Prestwich. Local Search and Backtracking vs Non-Systematic Backtracking. *AAAI Fall Symposium on Using Uncertainty within Computation*, Technical Report FS-01-04, AAAI Press, 2001, pp. 109–115.
17. S. D. Prestwich. Incomplete Dynamic Backtracking for Linear Pseudo-Boolean Problems. *Annals of Operations Research* 130:57–73, 2004.
18. S. D. Prestwich. Generalized Graph Colouring by a Hybrid of Local Search and Constraint Programming. *Discrete Applied Mathematics* (to appear).
19. S. D. Prestwich, S. A. Tarim, B. Hnich. Template Design under Demand Uncertainty by Integer Linear Local Search. *International Journal of Production Research* 44(22/15):4915–4928, 2006, Special Issue on Advances in Evolutionary Computation for Design and Manufacturing Problems.
20. L. Proll, B. M. Smith. Integer Linear Programming and Constraint Programming Approaches to a Template Design Problem. *INFORMS Journal of Computing* 10:265–275, 1998.
21. W. Ruml. Incomplete Tree Search Using Adaptive Probing. *17th National Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 2001, pp. 235–241.
22. D. Sabin, G. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. *11th European Conference on Artificial Intelligence*, 1994, pp. 125–129.
23. Y.-L. Shang, L.-S. Zhang. A Filled Function Method for Finding a Global Minimizer on Global Integer Optimization. *Journal of Computational and Applied Mathematics* 181(1):200–210, 2005.
24. C. Schulte, P.J. Stuckey. When do Bounds and Domain Propagation Lead to the Same Search Space. *ACM Transactions on Programming Languages and Systems* 27(3):388–425, 2005.
25. S. A. Tarim, I. Miguel. A Hybrid Benders' Decomposition Method for Solving Stochastic Constraint Programs with Linear Recourse. *Lecture Notes in Artificial Intelligence* 3978:133–148, Springer-Verlag, 2006.
26. J. P. Walser. Solving Linear Pseudo-Boolean Constraint Problems with Local Search. *14th National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, AAAI Press / MIT Press, 1997, pp. 269–274.
27. J. P. Walser, R. Iyer, N. Venkatasubramanyan. An Integer Local Search Method with Application to Capacitated Production Planning. *15th National Conference on Artificial Intelligence*, AAAI Press, 1998, pp. 373–379.
28. Y. Zhang, R. H. C. Yap. Arc Consistency on n-ary Monotonic and Linear Constraints. *6th International Conference on Principles and Practice of Constraint Programming, Lecture Notes In Computer Science* 1894:470–483, 2000.

An Evaluation of Dimension Reduction Techniques for One-Class Classification

Santiago D. Villalba and Pádraig Cunningham

School of Computer Science and Informatics, University College Dublin
{Santiago.Villalba,Padraig.Cunningham}@ucd.ie

Abstract. Dimension reduction (DR) is important in the processing of data in domains such as multimedia or bioinformatics because such data can be of very high dimension. Dimension reduction in a supervised learning context is a well posed problem in that there is a clear objective of discovering a reduced representation of the data where the classes are well separated. By contrast DR in an unsupervised context is ill posed in that the overall objective is less clear. Nevertheless successful unsupervised DR techniques such as Principal Component Analysis (PCA) exist – PCA has the pragmatic objective of transforming the data into a reduced number of dimensions that still captures most of the variation in the data. While one-class classification falls somewhere between the supervised and unsupervised learning categories, supervised DR techniques appear not to be applicable at all for one-class classification because of the absence of a second class label in the training data. In this paper we evaluate the use of a number of up-to-date unsupervised DR techniques for one-class classification and we show that techniques based on *cluster coherence* and *locality preservation* are effective.

1 Introduction

In recent years, the traditional distinction in machine learning between supervised and unsupervised techniques has been blurred due to the emergence of real-world problems that sit somewhere between these two extremes. In supervised classification problems, *discriminating* classifiers are trained using positive and negative examples. However, for a number of practical problems, counter-examples are either rare, entirely unavailable or statistically unrepresentative. Such problems include industrial process control, text classification and analysis of chemical spectra.

One-class classifiers (OCCs) have emerged as a set of techniques for situations where labelled data exists for only one of the classes in a classification problem. For instance, in industrial inspection tasks, abundant data may only exist describing the process operating correctly. It is difficult to gather training data describing the myriad of ways the system might operate incorrectly. A related problem is where negative examples exist, but their distribution cannot be characterised. For example, it is reasonable to provide characteristic examples of family pictures but impossible to provide examples of pictures that are

"typical" of non-family pictures. One-class classifiers are emerging as a solution, which *characterises* the target class, to distinguish it from all other classes.

In practice, one-class problems are typically of high dimension so DR is an important pre-processing step. Indeed the evaluation presented by Manevitz and Yousef [1] shows that one-class Support Vector Machine (SVM) performance is quite sensitive to the number of features used. This contrasts with two-class SVMs which are generally considered to be robust to high data dimensionality. This provides additional justification for DR in one-class classifier construction. However, the absence of counter-examples means it is difficult to identify a feature subset that encodes a *discriminating* description of the concept.

In this paper we review a range of unsupervised DR techniques and evaluate their performance on a number of OCCs. We find that DR based on *locality preservation* and *cluster coherence* principles seem particularly promising for OCC. However locality preservation is only effective when there are no irrelevant features in the full feature set; i.e. locality in the original space must be meaningful.

In the next section we provide an overview of OCC and describe the OCC techniques included in the evaluation. In section 3 we describe the DR techniques considered in the evaluation – the evaluation is presented in section 4. The paper concludes with a summary and some proposals for further research.

2 One-Class Classifiers

Traditionally machine learning tasks are divided into supervised and unsupervised categories. Roughly speaking, in unsupervised learning we are provided with a dataset (set of examples describing a real world concept) and the objective is to uncover some structure in the data. In supervised learning we are provided with a dataset where the information to be modeled is explicitly stated in the form of a label (a "class" label in the case of so called "classification problems") and the task is to predict the label for new (as yet unseen) examples.

One-class classification, also referred to as novelty or outlier detection, is sometimes thought of as a weaker form of supervised classification, where the only information we are given about the training examples is that they belong to the same class, arbitrarily called "positive" or "target". The task here is to accept or reject unseen examples depending on their similarity to the known positive examples. OCC approaches consequently can operate with very few, or no, negative training examples. In other words, one-class learning handles the "no-counter-example" and "imbalanced-data" problems by considering only positive examples. When unlabeled examples and/or small (or large) amounts of negative examples are available for training, several OCC techniques can also use them to fine-tune their performance.

In the current study we choose four different one class classifiers (Support Vector Data Description (SVDD), a k-Nearest Neighbours approach, a k-Means Clustering approach and a Gaussian Model), all of them available in the Data

Description toolbox [2], an open source Matlab software library of one-class classification tools.

Support Vector Data Description [3, 4]: The SVDD learns the hypersphere, defined by a center a and a radius R, that encloses (almost) all the training set while covering as little volume as possible. It can employ the kernel trick for learning more flexible boundaries, and the solution is found by solving a convex quadratic optimization problem analogous to the one found in Support Vector Machines.

Clustering (k-Means): Another approach to one-class classification is that of learning clusters, modeling the target class as a reduced set of cluster prototypes or centers onto which new examples are projected. Examples of clustering methods that can be used are the Self Organizing Map, Learning Vector Quantization or k-Means, the one we choose here [3, 5]. When a new example is to be classified, its distance to the nearest prototype is used to score the extent to which it is an outlier.

Lazy learning (k-Nearest-Neighbours): The Nearest Neighbour approach can be used for constructing one-class classifiers. The training data is stored and an *outlierness* criterion is calculated for new examples based on their nearest neighbours, i.e. their position relative to the seen examples. Several criteria have been proposed to measure the outlierness of an example [6, 7]. Here we use γ [6] which is the average of the distances to the k nearest neighbours.

Density estimation (Gaussian model) [3, 5]: The Gaussian model is a simple parametric one-class classifier which models the training data under the assumption that it comes from a unimodal multivariate normal distribution. These assumptions fit a lot of natural processes, but when they are violated this model introduces a large bias. The mean and the covariance matrix are estimated using an Expectation-Maximization approach and the Mahalanobis distance is used as the resemblance criterion.

We selected these OCC strategies and not others because of their conceptual simplicity, their well established properties and because we wish to explore the hypothesis that local learners are particularly well suited to one class problems. All bar SVDD are local learners.

3 Dimension Reduction Techniques

Research on dimension reduction has itself two dimensions. The first design decision is whether to select a subset of the existing features or to transform to a new reduced set of features. The other dimension in which DR strategies differ is the question of whether the learning process is supervised or unsupervised (see Figure 1). For OCC problems it seems that both feature selection and feature transformation strategies are relevant. However, given that labelled data is only available for one class, it seems that supervised DR techniques cannot be directly applied to OCC problems.

In supervised learning the objective of DR is to optimize the performance of the final system, that is, minimize the classification error. However, in one-

class classification performance estimation is difficult because the absence of counterexamples makes the estimation of the false positive rate hard. This makes it difficult also to tune the bias of the classifier and the best strategy to address this problem depends on the specifics of the data available.

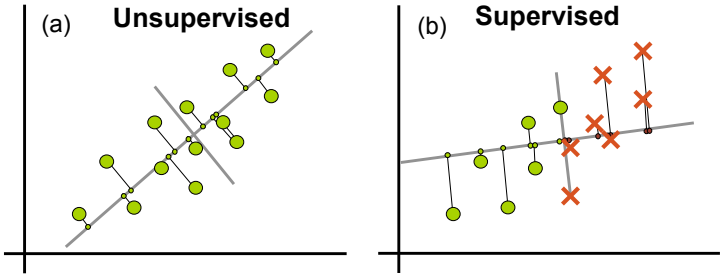


Fig. 1. In unsupervised DR (a) the "best" that can be done is to find a representation that maximises the variance in the data. When the data is labelled (b) a representation can be sought that improves the separation in the data.

A sensible approach is to try to synchronize the assumptions of both DR and classification. In our evaluation we consider four DR techniques; the first two are Principal Component Analysis and the Q-α algorithm presented by Wolf and Shashua [8]. The final two are based on the principle of *locality preservation* and these are described in section 3.1. We believe that locality preservation is of particular relevance to DR in the OCC domain because, usually one class classifiers rely on local neighbourhood relationships (see section 2).

Principal Component Analysis (PCA): PCA is the most commonly used technique for unsupervised dimensionality reduction [9]. It aims at finding the linear projections that best capture the variability of the data. In this study we use the common approach of keeping those directions that explains most of the variance. In [10] it is shown that retaining the high variance dimensions is not always optimal for one-class classification, so a minor components analysis (use the smallest variance directions) can be better under some circumstances.

The Q-α Algorithm: A well motivated criterion of cluster quality is cluster coherence, in graph theoretic terms this is expressed by the notion of objects within clusters being well connected and individual clusters being weakly linked. The whole area of spectral clustering captures these ideas in a well founded family of clustering algorithms based on the idea of minimising the *graph-cut* between clusters [11].

The principles of spectral clustering have been extended by Wolf and Shashua [8] to produce the $Q-\alpha$ algorithm that simultaneously performs feature subset selection and discovers a good partition of the data. As with spectral clustering, the fundamental data structure is the affinity matrix $\mathbf{A}$ where each entry $\mathbf{A}_{ij}$ captures the similarity (typically as a dot-product) between data points i and j. In order to facilitate feature selection the affinity matrix for $Q-\alpha$ is expressed as $\mathbf{A}_\alpha = \sum_{i=1}^{p} \alpha_i \mathbf{m}_i \mathbf{m}_i^\mathsf{T}$ where $\mathbf{m}_i$ is the i^{th} feature vector in the data matrix that has been normalised so to be centered on 0 and be of unit L_2 norm (this is

the set of values in the data set for feature i). $\mathbf{m}_i\mathbf{m}_i^\mathsf{T}$ is the *outer*-product of $\mathbf{m}_i$ with itself. α is the weight vector for the p features – ultimately the objective is for some of these weight terms to be set to 0.

In spectral clustering $\mathbf{Q}$ is an $n \times k$ matrix composed of the k eigenvectors of $\mathbf{A}$ corresponding to the largest k eigenvalues. Wolf and Shashua show that the relevance of a feature subset as defined by the weight vector α can be quantified by:

$$Rel(\alpha) = trace(\mathbf{Q}^\mathsf{T}\mathbf{A}_\alpha^\mathsf{T}\mathbf{A}_\alpha\mathbf{Q}) \tag{1}$$

They show that feature selection and clustering can be performed as a single process by optimising:

$$\max_{\mathbf{Q}\alpha} trace(\mathbf{Q}^\mathsf{T}\mathbf{A}_\alpha^\mathsf{T}\mathbf{A}_\alpha\mathbf{Q}) \tag{2}$$

subject to $\alpha^\mathsf{T}\alpha = 1$ and $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$.

Wolf and Shashua show that this can be solved by solving two inter-linked eigenvalue problems that produce solutions for α and $\mathbf{Q}$. They show that a process of iteratively solving for α then fixing α and solving for $\mathbf{Q}$ will converge. They also show that the process has the convenient property that the α_i weights are biased to be positive and sparse, i.e. many of them will be zero.

So the $Q - \alpha$ algorithm performs feature selection in the spirit of spectral clustering, i.e. it discovers a feature subset that will support a partitioning of the data where clusters are well separated according to a graph-cut criterion.

3.1 Locality Preservation

Locality preservation in dimensionality reduction techniques refers to the aim of keeping neighbourhood properties, e.g. objects that are close in the input space should also be close in the reduced space. Several linear and nonlinear techniques exploiting this criterion have recently been proposed. For the OCC problem it is rational to think that a locality preserving dimension reduction technique would be more practical in some cases than a global based one. Locality and density are frequently used in the OCC literature and both are present in the locality preservation bias.

Locality Preserving Projections (LPP): The idea behind LPP is that of finding subspaces which preserve the *local structure* in the data [12, 13]. Given a matrix $\mathbf{A}$ (symmetric, positive, invertible and, usually, sparse) which captures information about the relationships between the data points, for example the similarity in a neighbourhood, LPP finds the optimal linear embedding that respects the structure present in that matrix. LPP preserves cluster structures when clustering is based on locality, such as in the k-means algorithm, which renders an attractive quality for being used together with cluster analysis based OCCs. The details of LPP are described in Algorithm 1.

The embedding is defined by the bottom eigenvectors in the solution of equation 3. The construction of the weighted graph in the first and second steps of

Algorithm 1: LPP computation [12]

Construct the adjacency graph: let $\boldsymbol{S}$ be the training set and G denote a graph with $|\boldsymbol{S}|$ nodes. We put an edge between nodes i and j if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are "close". There are two variations:

- ε-neighbourhoods (parameter $\varepsilon \in \mathbb{R}$). Nodes i and j are connected if $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 < \varepsilon$ where the norm is the usual Euclidean norm in $\mathbb{R}^{dx}$.
- k nearest neighbours (parameter $k \in \mathbb{N}$). Nodes i and j are connected if i is among the k-nearest neighbours of j or viceversa.

Choose the weights for the graph edges: Here, as well, we have two variations for weighting the edges. $\mathbf{A}$ is a sparse symmetric $|\boldsymbol{S}| \times |\boldsymbol{S}|$ matrix with $\mathbf{A}_{ij}$ having the weights of the edge joining vertices i and j, and 0 if there is no such edge.

- Heat kernel (parameter $t \in \mathbb{R}$). When nodes i and j are connected put $\mathbf{A}_{ij} = e^{\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{t}}$
- Simple minded (no parameter). When nodes i and j are connected, put $\mathbf{A}_{ij} = 1$.

Eigenmaps: Compute the eigenvectors and eigenvalues for the generalized eigenvector problem:

$$\boldsymbol{S}\mathbf{L}\boldsymbol{S}^{\mathsf{T}}\boldsymbol{a} = \lambda \boldsymbol{S}\mathbf{D}\boldsymbol{S}^{\mathsf{T}}\boldsymbol{a} \tag{3}$$

where $\mathbf{D}$ is a diagonal matrix whose entries are column sums of $\mathbf{A}$, $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ji}$. $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix.

Algorithm 1 can be accomplished using any criterion. This permits more utilitarian approaches to be used by including *a priori* information about the problem.

Laplacian Score for Feature Selection (LS): The same criterion of locality preservation found in LPP can be applied in the feature selection context, where the merit of each feature is measured according to its locality preservation power [14].

As with $Q-\alpha$, there is no explicit enumeration of the feature subsets. Rather a nearest neighbour based graph is constructed from the training set and its spectrum is analysed to rank each variable. The first two steps of the algorithm are identical to those of LPP (Algorithm 1). For ranking each feature, its *Laplacian score* is computed. For the i-th feature we define:

$$\widetilde{\mathbf{m}}_i = \mathbf{m}_i - \frac{\mathbf{m}_i^{\mathsf{T}}\mathbf{D}\mathbf{1}}{\mathbf{1}^{\mathsf{T}}\mathbf{D}\mathbf{1}}\mathbf{1} \tag{4}$$

where $\mathbf{1} = [1, \cdots, 1]^{\mathsf{T}}$

The Laplacian Score (LS_i) for the i-th feature is:

$$LS_i = \frac{\widetilde{\mathbf{m}}_i^\top \mathbf{L} \widetilde{\mathbf{m}}_i}{\widetilde{\mathbf{m}}_i^\top \mathbf{D} \widetilde{\mathbf{m}}_i} \tag{5}$$

4 Evaluation

In this study we use the three biomedical datasets summarized in Table 1. One difficulty in assessing the performance of the combination of OCC and DR techniques on these datasets is the parameter optimisation (model selection) required for the different techniques. We followed a simple approach of fixing the values of the parameters to sensible values: $k = 6$, the number of clusters for k-means and neighbours for k-NN and the rest of the are left to the dd_tools default values. For computing the threshold of the resemblance function we set a 90% of training objects to be accepted (i.e. we consider that a 10% of the training examples are outliers).

Table 1. Summary of the three datasets used in the evaluation.

Dataset	n# Examples	n# Features	Target Class (n#)	Source
Bronchiolitis	118	22	1-Day (37)	[15]
Arrhythmia	452	279	Normal (245)	[16]
TIS-5%	668	927	TIS (178)	[17]

As parameters for the dimension reduction techniques, when applicable, we follow the principle of using the same parameters used in the couterpart classifiers. For example, the same value of k in the k-nearest neighbour is used when constructing the adjacency graph for LPP and LS, and the value of k in the k-means algorithm is set as the target number of clusters for Q-α. In both LPP and LS the "simple minded" weighting approach is followed. No further model selection is done and we also fix the rest of the parameters to *a priori* defined default values. The choice of target dimensionality is also an important issue. In this case we just explored all possibilities, from dimensionality 1 to the original dimensionality in feature selection or to the maximum defined by the feature transformation embedding.

In addition to the techniques described in section 3, we add two dimension reduction methods; a random feature selection process to provide a baseline and a supervised ranking of the features using information gain over the original multiclass datasets (this is "cheating"). In this way we try to establish if using a supervised feature selection criterion provides an upper bound for the accuracy of the dimension reduction system for one-class tasks.

The results are shown in Table 2. The Balanced Accuracy Rate, defined as the average of the true positive rate (sensitivity) and true negative rate (specificity),

is estimated by 10-fold cross validation. The figures shown are those obtained by the winning target dimensionality in each case (in parenthesis).

Table 2. Evaluation of the DR-OCC combinations. Balanced Accuracy Rate estimations for the winning dimensionality (in brackets) for each classifier / dimension reduction technique pair. In italics the *cheating* supervised feature selection. In **boldface** the unsupervised winning dimension reduction techniques for each classifier. It is clear that they are beneficial over the *No DR* case.

(a) Bronchiolitis

	No DR	IG	Random	Q-α	LS	LPP	PCA
Gauss	0.64(22)	*0.72(12)*	0.67(16)	**0.73(7)**	0.68(13)	0.72(13)	**0.73(17)**
KMeans	0.73(22)	*0.71(20)*	0.67(18)	**0.75(15)**	0.70(19)	0.70(15)	0.72(19)
KNN	0.66(22)	*0.70(16)*	0.66(21)	0.66(21)	0.68(21)	**0.72(12)**	0.65(20)
SVDD	0.65(22)	*0.72(1)*	0.68(19)	0.67(16)	**0.70(20)**	0.64(20)	0.67(16)

(b) Arrhythmia

	No DR	IG	Random	Q-α	LS	LPP	PCA
Gauss	0.55(279)	*0.78(57)*	0.71(83)	0.71(167)	0.70(167)	0.68(30)	**0.76(24)**
KMeans	0.68(279)	*0.77(28)*	0.70(195)	**0.74(167)**	0.68(279)	0.68(139)	0.72(20)
KNN	0.67(279)	*0.77(35)*	0.68(223)	**0.71(167)**	0.67(279)	0.68(139)	0.70(20)
SVDD	0.68(279)	*0.75(83)*	0.68(167)	**0.70(167)**	0.66(279)	0.65(167)	0.69(29)

(c) TIS

	No DR	IG	Random	Q-α	LS	LPP	PCA
Gauss	0.54(927)	*0.83(3)*	0.54(463)	**0.82(27)**	0.70(18)	0.54(185)	0.77(4)
KMeans	0.45(927)	*0.79(3)*	0.49(1)	0.67(18)	0.56(6)	0.52(3)	**0.74(4)**
KNN	0.45(927)	*0.81(5)*	0.49(4)	0.70(19)	0.55(19)	0.54(3)	**0.77(2)**
SVDD	0.38(927)	*0.82(2)*	0.49(32)	0.69 (3)	0.59 (1)	0.49 (3)	**0.72(2)**

In Table 2 dimension reduction is beneficial in all cases. This is not surprising since we chose datasets that require dimension reduction to achieve good results in a supervised setting. In most cases the supervised DR technique provide an upper bound for the performance. Q-α is very promising, it gives high scores to relevant features and yields consistent improvements.

In the case of the Arrhythmia and especially the TIS dataset the locality preserving principle of LPP is not competitive with the rest of the unsupervised criteria. This is due to the fact that the presence of a lot of irrelevant features in the full feature set renders locality in that space inappropriate. Further, when used with a non-local learner such as SVDD, LPP compares adversely even with the random feature selection criterion.

In Figure 2 we show the evolution of the sensitivity / specificity tradeoff for two dataset / classifier combinations: k-means in TIS and k-NN in Bronchiolitis. For locality-based classifiers it usually holds that the more the dimensionality is

reduced, the better the sensitivity and the worse the specificity. This is due to the fact that computing description of objects in low dimensional spaces is easier while discriminative power is lost, so descriptions also capture outliers. When the dimensionality is high the descriptions become inaccurate and so the classifiers become accept-all or, more often, reject-all machines. This phenomenon is more significant in the case of LPP.

Fig. 2. Evolution of Sensitivity (up) vs Specificity (bottom) tradeoff with increasing target dimensionality. On the left applying k-NN to the bronchiolitis dataset. On the right, applying k-means to the TIS dataset with dimensionality varying from 1 to 59. (This figure is best viewed in colour).

5 Conclusions

This paper reports progress in research on the applicability of DR techniques (specifically techniques from unsupervised learning) for OCC problems. At this stage we feel there are two key findings:

- We have demonstrated the potential improvements to be had by applying carefully selected DR techniques prior to one-class classification.
- In some circumstances techniques based on *cluster coherence* and *locality preservation* are particularly effective. We believe that LP-based techniques are appropriate when there are few irrelevant features in the data set, i.e. in

order for locality to be meaningful in the original feature space it is necessary for most if not all of the features to be relevant.

As already stated, *locality preservation* seems an appropriate criterion for OCC tasks but it contains the implication that none of the input features are irrelevant; they may be just redundant. The key issue here is how meaningful the distance functions are. We encounter the paradoxical situation that for reducing the dimensionality of the data one needs to rely on distance measures which, most probably, are not meaningful in the original high dimensional space. How to learn a proper metric from the data itself instead of imposing a pre-specified one is an active research field in several areas of classification. For one-class classification, once again, the current techniques are not directly applicable because they use information from both sides of the classification boundary. However, related techniques could lead to useful one-class metric learning techniques.

References

1. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. Journal of Machine Learning Research **2** (2001) 139–154
2. Tax, D.M.J.: DDtools, the Data Description Toolbox for Matlab (April 2007)
3. Tax, D.M.J.: One-class classification. Concept learning in the absence of counterexamples. PhD thesis, Delft University of Technology (2001)
4. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Mach. Learn. **54**(1) (January 2004) 45–66
5. Juszczak, P.: Learning to recognise, a study on one-class classification and active learning. PhD thesis, Delft University of Technology (2006)
6. Harmeling, S., Dornhege, G., Tax, D., Meinecke, F., Muller, K.R.: From outliers to prototypes: Ordering data. Neurocomputing **69**(13-15) (August 2006) 1608–1618
7. Rieck, K., Laskov, P.: Language models for detection of unknown attacks in network traffic. Journal in Computer Virology **2**(4) (February 2007) 243–256
8. Wolf, L., Shashua, A.: Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. Journal of Machine Learning Research **6** (2005) 1855–1887
9. Jolliffe, I.T.: Principal Component Analysis. Springer (October 2002)
10. Tax, D., Muller, K.R.: Feature extraction for one-class classification. In: ICANN/ICONIP 2003, Springer Berlin / Heidelberg (2003) 342–349
11. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems. Volume 14. (2001)
12. He, X., Niyogi, P.: Locality preserving projections. In: NIPS: Advances in Neural Information Processing Systems. (2003)
13. He, X.: Locality Preserving Projections. PhD thesis, University of Chicago (2005)
14. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: NIPS: Advances in Neural Information Processing Systems. (2005)
15. Doyle, D.: A Knowledge-Light Mechanism for Explanation in Case-Based Reasoning. PhD thesis, University of Dublin, Trinity College (2005)
16. Bay, S.D., Hettich, S.: The UCI KDD archive [http://kdd.ics.uci.edu] (1999)
17. Liu, H., Wong, L.: Data mining tools for biological sequences. J Bioinform Comput Biol **1**(1) (April 2003) 139–167

Computational Modelling of Switching Behaviour in Repeated Gambles

Jiaying Zhao[1], and Fintan Costello[1]

[1]School of Computer Science and Informatics,
University College Dublin, Belfied, Dublin 4, Ireland.

Abstract. In this paper we present a computational model which predicts people's switching behaviour in repeated-choices gambling scenarios. Specifically, the model suggests that people switch away from an option due to the amount and the probability of losses experienced compared to wins, and also due to the number of decisions people have made consecutively in that option. Results obtained so far suggest that our model gives more accurate predictions than the previous Bayesian-EU model and expectancy-valence model.

Keywords: Switching, Repeated Gambles, Iowa Gambling Task.

1 Introduction

Making a good decision is crucial to human survival. Most studies on decision making have typically used single-choice scenarios. For example, the decision maker has to choose either winning $500 for sure or a 0.5 chance of winning $1,000 [1]. However, in real life people often make a number of choices. By making repeated choices they tend not to stick with one option but to switch between available choices [2,3]. For example, people switch between different stocks and choose the one that gives the best return. By investigating such switching behaviour we can offer a more in-depth view of the decision process involved in making repeated choices.

In this paper we attempt to clarify the following questions: what causes us to switch between choices; and what factors are involved in such switching, by modelling people's switching behaviour in a gambling task. The study will examine task factors (e.g. contingencies of wins and losses) in people's switching behaviour in the task. The Iowa Gambling Task (IGT) designed by Bechara and colleagues [4] is used for our investigation, as firstly the task has been well established in the decision literature and secondly it provides a paradigm for investigating switching behaviour in repeated choices. The main interest of the current paper is to focus on the switching behaviour of healthy participants.

2 The Iowa Gambling Task

The IGT paradigm resembles real-world contingencies [4]. The task consists of four decks of cards (labelled as A, B, C, and D), each containing 40 cards. The participant is asked to choose a total of 100 cards from the four decks and to maximise their gain. The participant can select any deck and is free to switch among the decks at any time. The payoff schedule for each deck is pre-arranged but is not explicitly told to the participant.

Every card in the four decks gives a definite win but some cards also give an unpredictable loss. In deck A, every card gives a definite $100 win but there are 50% of the cards, each of which also gives an average $250 loss. This means that the average net payoff per card in deck A is a $25 loss (0.5*100-0.5*(250-100)). In deck B, every card gives a definite $100 win but there are 10% of the cards, each of which also gives a $1250 loss. This means that the average net payoff per card in deck B is a $25 loss (0.9*100-0.1*(1250-100)). In other words, decks A and B are disadvantageous as choosing decks A and B would eventually lead to an overall loss in the long run. In deck C, every card gives a definite $50 win and there are 50% of the cards, each of which also gives an average $50 loss. Thus, the average net payoff per card in deck C is a $25 gain (0.5*50-0.5*(50-50)). In deck D, every card gives a definite $50 win and there are 10% of the cards, each of which also gives a $250 loss. That is, the average net payoff per card in deck D is a $25 gain (0.9*50-0.1*(250-50)). In other words, decks C and D are advantageous since choosing decks C and D would lead to an overall gain in the long run.

A typical finding reported with this task is that healthy control participants gradually learn to favour the advantageous decks, but people with orbitofrontal frontal cortex damage persist in choosing the disadvantageous decks [4,5,6]. Damasio [7] proposes somatic marker hypothesis to account for the results. It postulates that somatic markers that are bodily reactions to environmental stimuli trigger and associate emotions with the outcomes obtained from previous experiences through learning, and they serve as an alarm bell that leads to the rejection of the negative course of action but as an incentive that promotes the pursuit of the positive course of action. In other words, negative emotions are associated with the losses generated by the disadvantageous decks and people resist future choices in those decks as an avoidance of negative feelings; positive emotions are linked with the gains produced by the advantageous decks and thus direct more choices to be made in those decks as a pursuit of happy feelings.

According to this hypothesis, one would expect that after a loss negative emotions (e.g. sadness) will occur which trigger(s) people to switch away from the deck and after a gain positive emotions (e.g. happiness) will occur which cause(s) them to stay at the deck. However, this might not always be the case. For example, we observed from Bishara and colleagues' [8] data on healthy participants that people do not necessarily switch away from a deck after a loss occurs. Specifically, we found that people tend to stay at the advantageous decks even though a loss has occurred (around 76% of the times they stayed after getting a loss), and people tend not to stay at the disadvantageous decks when a loss occurs (around 12% of the times they stayed after a loss). Interestingly, we also observed that 55% of the times that they switch away from a deck are when no loss occurs on the previous choice. In other words,

more than half of the times people switch away from a deck when no loss occurs previously.

If no loss occurs why do people switch away? There are several explanations. Some authors have indicated that people making repeated choices among a limited set of alternatives often switch away from optimal options to other options from which they expect to derive less pleasure [9,10]. Levine, Mills, and Estrada [11] suggest that people might choose different decks occasionally to 'try their luck', or switch to other decks simply due to fatigue or boredom. We postulate that the reason for switching away from a deck with no loss involved is the fear or nervousness of obtaining a loss on the next choice. For example, a participant who has won on five consecutive cards in a deck may believe that a loss is more likely on the sixth card, and therefore he (she) might switch away. We suggest that, the longer people have stayed at a deck, the more afraid (nervous) they are about getting a loss on the next card in that deck, thus the more likely they are to switch away from that deck. Therefore, the reason for switching away is mainly due to two factors: one is obtaining a loss, and the other is the fear or nervousness of obtaining a loss. We also postulate that if people were to switch away from a deck, they would switch to a deck which gives the largest wins and also the highest probability of winning.

3 Cognitive Models of the IGT

To the best of our knowledge, only a handful of cognitive models have been developed to account for the results in the IGT to date [11,12,13]. Among those, even fewer are developed to account for people's switching behaviour. Previous models primarily focus on people's overall preference for specific decks but ignored the exact details of how people choose and how they switch among the four decks. In this paper we argue that the factors responsible for switching are the contingencies of wins and losses and the fear or nervousness of getting a loss. Before presenting our model, two prominent models, Bayesian-expected utility model and expectancy-valence learning model [12], are examined first which serve as a basis for our model.

3.1 Bayesian-Expected Utility Model

The Bayesian-expected utility (EU) model operates under the principle that people use bounded rational decision strategies to make their choices [12]. Details of the model are not described here (see Busemeyer and Stout [12]). The model claims that a deck should be chosen if it has the maximum EU. This is conducted in several steps. Firstly, the probability of losses is calculated for each deck chosen; secondly, the utility of the net payoff of each choice is calculated; thirdly, the expected utility of a deck is computed; and finally, the deck should be chosen on the next pick if it has the maximum EU. If the preferred deck is exhausted then the deck with the next highest EU will be chosen.

This model is adequate at capturing the rational aspect of people's choices. That is, a win increases the EU of a deck and suggests that people should stay with

that deck. In contrast, a loss decreases the EU of the deck and suggests people might switch to another deck if it has the maximum EU.

However, there are two limitations with this model. Firstly, if a large loss is encountered (for example, in deck B), the EU decreases dramatically and it may drop to so small that the deck may be unlikely to be chosen again. However, people still choose deck B quite often despite a previous large loss. Levine et al. [11] suggest that people's preference of deck B is due not only to the certainty effect (there is 90% chance of winning in deck B), but also to their tendency to underweight rare events [14]. The Bayesian-EU model fails to capture this aspect of people's choices. Secondly, in terms of switching behaviour this model only predicts that the deck with the maximum should be chosen. It cannot explain the situations when people choose a deck which does not have the maximum EU, for example, when people switch away from a deck when no loss occurs.

3.2 Expectancy-Valence Learning Model

Details of the expectancy-valence learning model are not described here (see Busemeyer and Stout [12]). The model assumes that people integrate the wins and losses experienced on each pick into a single affective reaction called a valence. The expected valence for a deck is updated by an adaptive learning mechanism. The choice made on each pick is a probabilistic function of the expected valence associated with each deck, which is an increasing function of the expectancy for that deck and a decreasing function of the expectancies for the other decks.

If the preferred deck is exhausted then the deck with the next highest probability will be chosen. According to Busemeyer and Stout [12], this model gives the best fit for the IGT data. They find that the model can provide the best match to the observed data in terms of the proportion of people's choices made from advantageous decks. This model has similar limitations to the Bayesian-EU model: if a large loss is encountered, the expected valence drops significantly, and so does the probability of choosing that deck, thus the deck may be never chosen again; and it cannot capture people's switching tendency when no loss is involved. Both of the models give logical explanations but do not necessarily reflect how people choose.

3.3 Assessing the Models

Since both models operate upon certain learning mechanisms, it is safe to assume that neither of the models can work without seeing a number of choices first. Thus, the models cannot make predictions without having a certain amount of experiences.

A fair way to assess the cognitive models would be to give the models the first n choices and then ask the model to predict the n+1 choice. In this way the models makes the prediction based on the same experiences shared with the decision maker. The n can vary from 1 to N-1 (N is the total number of cards to be picked) and the model can at most make N-1 predictions. The prediction accuracy is calculated as the number of accurate predictions divided by the total number of predictions.

The two models were tested on the set of data obtained by Bishara and colleagues [8] on 31 healthy participants using a modified IGT task in which a total of 120 cards were chosen. A JAVA program was written to conduct the assessment of the two models. The models were tested for each participant.

It was found that the Bayesian-EU model gave an average of 47.13 (SD=20.15) accurate predictions out of 119 total predictions made (see Fig. 2). The accuracy was 0.396 which means that the Bayesian-EU model predicted accurately 39.6% of the times. The expectancy-valence learning model gave an average of 48.45 (SD=21.93) accurate predictions out of 119 predictions. The accuracy was 0.407 which means that the expectancy-valence learning model predicted accurately 40.7% of the times. Since the criterion for judging accuracy is 25% (1 in 4 decks), the model prediction accuracy for each participant was compared to 0.25. It was found that the Bayesian-EU prediction accuracy was significantly higher than 25% ($t=.803$; $df=30$; $p<0.05$); and the expectancy-valence learning model prediction accuracy was also was significantly higher than 25% ($t=4.748$; $df=30$; $p<0.05$). This means that both models could predict more accurately than random guessing.

4 A Computational Model of Switching Behaviour

As seen the Bayesian-EU model and the expectancy-valence learning model were relatively inadequate at predicting the choices people made. We propose a computational model which examines task factors in people's choices in the IGT task. More importantly, our model predicts whether people should switch away from or stay at a deck at a given point of the task, and it also predicts if switch, to which deck people switch. There are two task factors that determine switching. These two factors are calculated mathematically. Factor one (F_1) describes the task factor which includes the contingencies of winning and losing and factor two (F_2) represents the length of time people have stayed in a deck.

4.1 Task Factors

For F_1 it is assumed that the probability of switching away is an increasing function of the probability of losing multiplied by the amount of losses and a decreasing function of the probability of winning multiplied by the amount of wins. The estimated probability of a loss is the same as the one in the Bayesian-EU model, which is an estimated probability $P_D(t)$ given that deck D is chosen on trial t using a beta prior distribution updating rule:

$$P_D(t) = \frac{f_D(t) + f(0)}{n_D(t) + n(0)} . \quad (1)$$

where $f_D(t)$ is the number of cards producing a loss experienced by choosing deck D up to and including trial t; $n_D(t)$ is the total number of trials that deck D was chosen; and $f(0)$ and $n(0)$ reflect the prior estimates before any experience.

The amount of losses $L[D(t)]$ is instantiated as the average of total losses experienced by choosing deck D up to and including time t:

$$L[D(t)] = \frac{\sum_{j=1}^{N} L[D]}{N} . \quad (2)$$

where N is the total number of cards chosen so far in deck D. The probability of winning is simply 1- $P_D(t)$, which means the probability of cards producing a net win. The amount of wins $R[D(t)]$ is instantiated the same way as the losses, which is the average of total wins experienced by choosing deck D up to and including time t:

$$R[D(t)] = \frac{\sum_{j=1}^{N} R[D]}{N} . \quad (3)$$

where N is the total number of cards chosen so far in deck D. Therefore, F_1 is denoted:

$$F_1 = \frac{P_D(t) \times L[D(t)]}{(1-P_D(t)) \times R[D(t)]} . \quad (4)$$

We predict that F_1 will be positively correlated with the switching probability. That is, the larger the probability of losing and the average losses of a deck, and the smaller the probability of winning and the average wins of the deck, the more likely people are to switch away from that deck on the next choice. If people do switch (out of their own will or if the preferred deck is exhausted), they would choose the deck with the maximum (1- $P_D(t)$) × $R[D(t)]$.

F_2 is denoted as n/N, where n is the number of continuous picks in a deck and N is the number of total picks in a deck. It is assumed that F_2 will be positively correlated with the switching probability, since the longer people stay in a deck consecutively, the more likely they are to switch away from that deck. The probability of switching away from a deck is a multiple regression predictor equation computed as follows. Multiple regression produces the best fit with F_1 and F_2 as the predictor variables, and switching probability as the dependent variable.

$$S = a \times F1 + b \times F2 . \quad (5)$$

4.2 Assessing the Model

The switching probability was calculated as a probability function which was the number of switches in every *N* choices (*N* could be from 2 to the total number of picks, and in our case *N*=10). For example, if there were 7 times of switching in the first 10 picks, the switching probability of the first 10 picks was 0.7. The values of F_1 and F_2 were calculated for each pick for each participant. To match the switching probability data we averaged the values of F_1 and F_2 in every *N* choices.

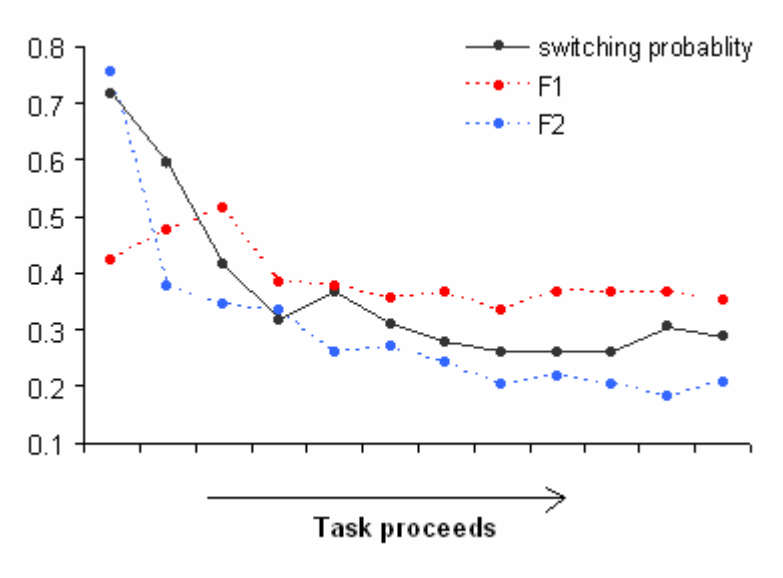


Fig.1. The graph shows the switching probability and the values of F_1 and F_2 as the task proceeds. The *x* axis shows the task process in which the 120 choices in the IGT are broken down to 12 blocks of 10 choices.

In Fig. 1 we can see that the switching probability decreased in the task, which means that as people have more experiences with the four decks they gradually develop a preference for a specific deck(s), therefore they do not switch as much as they used to. The values of F_1 and F_2 followed the same trend as the switching probability. Thus, we had good support that the coefficients *a* and *b* would both be positive.

We then ran multiple regression across all individual participants in the data of Bishara and colleagues [8]. The assessment was conducted using a JAVA program. The two coefficients were compared to 0 using one-sample t-test. We found that the mean of coefficient a was 0.146 and was significantly above 0 ($t=2.456$; $p<0.05$). We also found that the mean of coefficient b was 0.151 and was also significantly above 0 ($t=2.243$; $p<0.05$).

We further tested the model using the next-choice-prediction method. The means of the coefficients (a=0.146, b=0.151) were put into the equation: $S = a \times F_1 + b \times F_2$. We set the threshold for switching to 0.5, arbitrarily. This means that if the S value was below 0.5, then the model should stay with the same deck as before; if the S value was equal to or greater than 0.5, the model should switch to another deck with the maximum $(1- P_D(t)) \times R[D(t)]$. The threshold for switching was also tested, using 0.1, 0.3, 0.5, 0.7, 0.9, and we found that 0.5 gave the highest prediction accuracy.

Using the next-choice-prediction method it was found that the average correct predictions were 75.90 (SD=18.16) out of 119 predictions, which means that our model could predict accurately people's choices at 63.8% of the times. It was found that the accuracy was significantly higher than 25% (t=14.148; df=30; $p<0.01$).

Figure 2 shows the prediction accuracy of our model and competing models during the task. In general it showed that the accuracy increased as the task proceeded. Furthermore, it could be seen that after some point in the middle of the task, the accuracy persisted above 0.7. This means that after almost half of the IGT task, the model could predict people's choices with 70% accuracy, which is well above the random criterion (25%). Our model prediction accuracy for each participant was compared to those of the two competing models, and it was found that our model gave significantly more accurate predictions than the other models (F=20.169; df=2; $p<0.001$). Thus, we concluded that our model was superior to the pervious models.

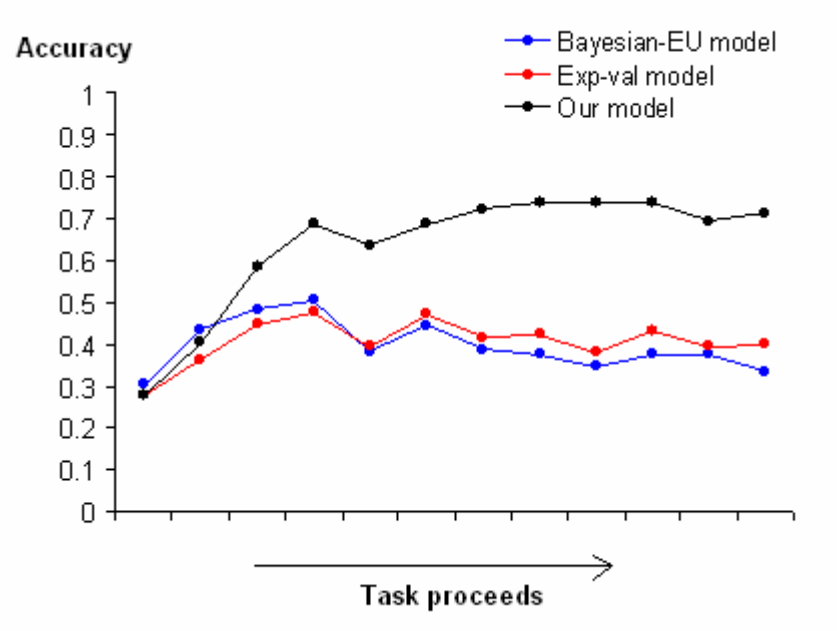


Fig.2. The prediction accuracy for the three models as the task proceeds. The x axis shows the task process in which the 120 choices in the IGT are broken down to 12 blocks of 10 choices.

5 Discussion

Results obtained so far suggest that our computational model fares better than the previous Bayesian-EU model and expectancy-valence model. Our model provides an account of how and why people switch among their choices. Specifically, it claims that people switch away from an option due to the amount and the probability of losses experienced compared to wins (F_1), and also due to the number of decisions they have made consecutively in that option (F_2). To our knowledge no model so far has included a factor that represents the number of decisions people make in an option. In our model F_2 gives an account of people's tendency to switch away from a deck even though no loss occurs previously.

Although our model can predict with 63.8% accuracy on average, which is much higher than the previous models, there is still more than 30% chance that our model cannot predict. We suggest that the remaining percentage might be due to the randomness inherent in people's choices, especially during the beginning of the task when people have no idea about their options.

Previously we suggest that the longer people have stayed at a deck, the more afraid (nervous) they are about getting a loss in that deck. This idea is not tested in the current analysis, since with the available data we have no access to people's emotional states during the task. Thus, for future work we suggest that the emotional states can be examined. In fact, we propose that there are two emotional factors (E_1 and E_2), each of which corresponds to the task factors (F_1 and F_2). Another experiment on the IGT is to be conducted with two purposes: firstly, our model can be tested in this experiment; secondly, the two emotional factors can be examined.

E_1 represents the level of happiness after choosing a card. We assume that the happier people are about their choices, the less likely they are to switch away. E_2 represents the level of nervousness or fear after choosing a card but before seeing the results of that card. We assume that the more nervous or afraid people are about their choices, the more likely they are to switch away from that deck.

It should be pointed out that our model aims to explain people's switching behaviour in the IGT, but not to provide an algorithm for making advantageous choices. Past research on the IGT has shown that people unanimously prefer the two advantageous decks. Our model can predict people's choices relatively accurately, thus it should also generate a preference for the advantageous decks.

6 Conclusion

We have presented a computational model which examines two task factors in people's switching behaviour in the IGT task, and claims that people switch away from a deck not only because of the characteristics of the task (contingencies of wins and losses), but also because of the number of choices they have made continuously in the deck. Using the next-choice-prediction method, it is found that our model predicts with 63.8% accuracy on average, which is much higher than the previous models. Our model and the proposed emotional factors will be further examined in the forth-coming experiment.

References

1. Kahneman, D., Tversky, A.: Prospect Theory: An Analysis of Decisions under Risk. Econometrica, 47, 263--291 (1979)
2. Coombs, C.H., Meyer, D.E.: Risk-preference in Coin-toss Games. J. Mathematical Psychology, 6, 514--527 (1969)
3. Keren, G., Wagenaar, W.A.: Violation of Utility Theory in Unique and Repeated Gambles. J. Experimental Psychology: Learning, Memory and Cognition, 13, 387--391 (1987)
4. Bechara, A., Damasio, H., Tranel, D., Anderson, S.W.: Insensitivity to Future Consequences Following Damage to Human Prefrontal Cortex. Cognition, 50, 7--15 (1994)
5. Bechara, A., Damasio, H., Damasio, A.R., Lee, G.P.: Different Contributions of the Human Amygdala and Ventromedial Prefrontal Cortex to Decision-making. J. Neuroscience, 19, 5473--5481 (1999)
6. Bechara, A., Damasio, H., Tranel, D., Anderson, S.W.: Dissociation of Working Memory from Decision Making within the Human Prefrontal Cortex. J. Neuroscience, 18, 428--437 (1998)
7. Damasio, A.R.: Descartes' Error: Emotion, Reason, and the Human Brain. Grosset/Putnam, New York (1994)
8. Bishara, A.J., Pleskac, T.J., Fridber, D.J., Yechiam, E., Lucas, J., Busemeyer, J.R., Fin, P.R., Stout, J.C.: Models of Risky Decision-making in Marijuana and Stimulant Users. Unpublished Manuscript (2006)
9. Kahn, B.E., Ratner, R.K., Kahneman, D.: Patterns of Hedonic Consumption over Time. Marketing Letters, 8, 85--96 (1997)
10. Ratner, R.K., Kahn, B.E, Kahneman, D.: Choosing Less-preferred Experiences for the Sake of Variety. J. Consumer Research, 26, 1--15 (1999)
11. Levine, D.S., Mills, B., Estrada, S.: Modeling Emotional Influences on Human Decision Making under Risk. In: Proceedings of Internal Joint Conference on Neural Networks, pp.1657--1662, IEEE Press, Montreal (2005)
12. Busemeyer, J.R., Stout, J.C.: A Contribution of Cognitive Decision Models to Clinical Assessment. Psychological Assessment, 14, 253--262 (2002)
13. Fum, D., Stocco, A.: Memory, Emotion, and Rationality: An ACT-R Interpretation for Gambling Task Results. In: Schunn, C.D., Lovett, M.C., Lebiere, C., Munro, P. (eds.). Proceedings of the Sixth International Conference on Cognitive Modelling, Lawrence Erlbaum, pp. 211--216. Mahwah, NJ (2004)
14. Barron, G., Erev, I.: Small Feedback-based Decisions and Their Limited Correspondence to Description-based Decisions. J. Behavioral Decision Making, 16, 215--233 (2003)

Recurrent Progressive Deepening with Pruning.

Arthur W. S. Cater

School of Computer Science and Informatics, UCD Dublin, Belfield, Dublin 4.

arthur.cater@ucd.ie

Abstract.

Progressive deepening (PD; also known as Iterative deepening), a common variant of Minimax search, can readily be adapted to search of AND/OR trees using 3-valued logic where terminal nodes may have values 'True' and 'False' and non-terminal nodes may also have value 'Unknown'. When an interior OR or AND node has a daughter node with value determined as True or False respectively, its remaining daughters need not be evaluated ("pruning"). Unknown values arising when the search is too shallow to reach terminal nodes also allow pruning at interior nodes. In game trees with generally good move ordering this is likely to save considerable effort especially at AND nodes. The paper shows how such pruning can cause difficulties in PD in the specific case where an AND node's daughter with a previously Unknown value acquires a True value, especially when using a transposition table to record results of subtree searches. The impact can be managed by the novel but simple 'Recurrent Progressive Deepening' technique. The finding generalizes to the more common case typical of alpha-beta search where evaluations are numeric rather than Boolean.
Key words: Iterative deepening, and/or game trees, pruning.

§1. Introduction

Minimax is the most basic algorithm used in searching large game trees, such as those that arise in play of Chess, Go, Draughts, Othello, and many other two-player games. Many variations of Minimax are commonly used, notably alpha-beta pruning. Progressive Deepening (PD; also known as Iterative Deepening) is a well established technique [1] useful with basic Minimax, with or without alpha-beta, and with or without numerous other refinements and variations. PD involves conducting the Minimax (or other: see [2]) search in depth-first fashion to some shallow depth, then – if time allows and no move is incontrovertibly best – repeatedly searching to ever greater depths. At first sight this seems wasteful since the top levels of the tree are regenerated over and over again at successive iterations. However the wasted effort is typically only a small fraction of the overall effort, and there are three notable compensating advantages which make it well worth while. The waste is small because

the branching factor b is typically quite high[1] and so the effort wastefully expended in searching to some depth d is around $1/b$ of the effort at depth $d+1$. With this investment three advantages may be won. First, it is possible to find shortest winning sequences as with breadth-first search, but without its exponential memory requirement. Second, it is possible to have flexible time control for tournament situations, since the approximate results of a fairly deep search are available if a still deeper search cannot be completed in time. Third, the principal variation found by shallower searches can be used to order moves in the upper part of the game tree in a manner approximating best-move-first, which pays huge dividends when pruning techniques such as alpha-beta are also used [3].

Game tree search is usually concerned with finding 'the best move'. Since in many interesting games the game tree is so large as to be in practice inexhaustible, it is normal to use Minimax with an evaluation function that produces a numerical estimate of the goodness of a position from some player's perspective: high values correspond to positions good for the "maximizing player". However it is sometimes useful for a game-playing program to search instead for a sequence of moves to guarantee some specific objective against any defence. In such cases the game tree is more like an AND/OR tree. One player, the "achiever", is satisfied to find one move that cannot be refuted[2]: positions arising from his possible moves are grouped under an OR node. The opponent is overcome if all his moves allow the objective to be attained: positions arising from his moves are grouped under an AND node.

Programs for playing Go commonly search this type of tree for achieving an objective, partly because it is extraordinarily difficult to fashion a sufficiently reliable whole-board evaluation function for that game [4]. Figure 1 shows an extract from an example position (on a small board), where a program may wish to determine whether the black player could accomplish a goal of separating certain white stones. Programs for playing Chess, which is likely familiar to more readers, seldom search such trees and seldom pose themselves that kind of objective. Nevertheless, consider the situation of an advanced passed White pawn in Chess: can it be promoted to a Queen? See Figure 2. This kind of question could be answered by searching a tree of possible White moves and possible Black counter moves. At any node where the pawn has been promoted to a Queen, the valuation is True; at any node where the pawn has been captured or either king checkmated or stalemated, the valuation is False. At any other node, the valuation depends on the values of daughter nodes combined appropriately. If the search is not deep enough the value is Unknown. The technique of Proof Number Search [5] is employed in numerous game playing programs for searching such trees. The present paper is concerned however with depth-first search enhanced with both progressive deepening and pruning.

[1] In Chess the branching factor b is estimated at around 35; in full-size Go it is around 200 for whole-board searches; in Arimaa it is in the thousands.

[2] The player may in practice wish to select among several irrefutable moves if more than one exists. That issue is beyond the scope of this paper.

Good move ordering is known to be beneficial for speeding search with alpha-beta pruning [3]. Perfect move ordering is unachievable; if it were possible no search would be needed. In AND/OR trees the three values {True, Unknown, False} can be ordered exactly as numbers can be, and used by a trivial variation of alpha-beta pruning. Good move ordering for the achiever will favour those moves which lead to the goal being not only achieved, but achieved in the fewest steps against strongest opposition. Good move ordering for the opponent will favour any moves which deny the goal, or when none exist, those that delay the goal as long as possible. When search is being conducted to some horizon therefore, the effect of perfect move ordering for the opponent is to place the successor nodes of OR nodes so that successors provably False (if there are any) precede those Unknown, which precede those provably True (all proofs being within the horizon). Good move ordering will approximate this pattern.

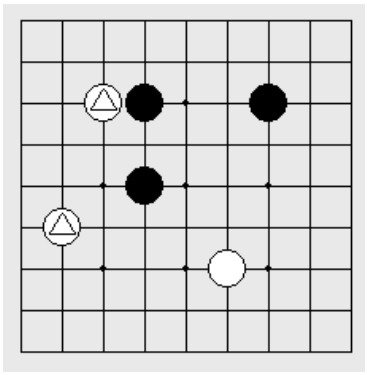


Figure 1:
Go, Black to play. Can the connection of the marked stones be prevented?

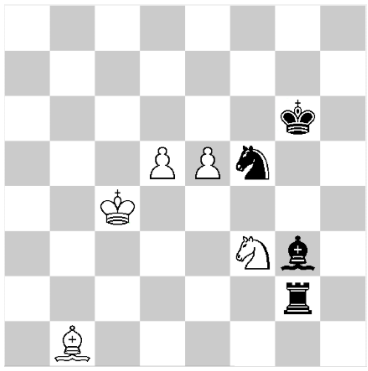


Figure 2:
Chess, White to play. Can the pawn on d5 be promoted to a queen?

Searching deeper (as in PD) may resolve an Unknown result into a True or False. §2 shows that this can result in a burst of activity in what will be termed 'First Order Progressive Deepening' or 'FOPD'. §3 explains the difficulty this causes, and how it may be easily resolved by a refinement called 'Recurrent Progressive Deepening' or 'RPD'. §4 discusses refinements applicable both to FOPD and RPD, namely the established use of Transposition Tables to bypass goal evaluations, and the less established notion of deepening schedules. §5 offers conclusions, particularly the claim that RPD is just as applicable and just as desirable in conventional numeric alpha-beta as it is in the special case of AND/OR game trees with three-valued logic.

§2. Burst of activity

Figure 3 shows an example of a partially evaluated and/or tree (unrelated to the Chess or Go questions of figures 1 and 2, and with generally low branching factor for the sake of a small size of figure). Layers consisting of 'Or' nodes alternate with layers consisting of 'And' nodes. 'And' nodes have a bar across the downward links to their successors. Leaf nodes that have a definite 'true' or 'false' value are shown as T or F respectively; interior nodes which acquire a 'true' or 'false' value by virtue of the subtrees they dominate are shown as '*t*' or '*f*' respectively; and nodes whose value is not yet known are shown as 'u'. Several of these unknown nodes may stay unknown, because there is already sufficient information to assign 'true' or 'false' to a parent 'Or' or 'And' node respectively. That is, the search of those subtrees may be pruned.

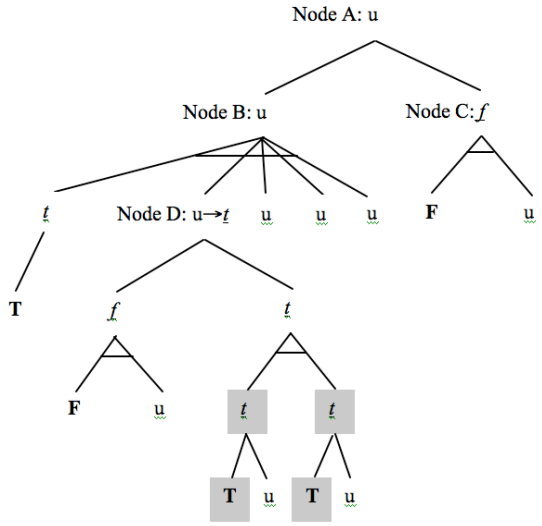


Figure 3:
Example of partially evaluated and/or tree.

Pruning is beneficial because it saves the generation and evaluation of whole subtrees of a game tree. The savings can be dramatic. For simplicity imagine a typical branching factor b at interior nodes, and a typical number p of the daughter nodes of an interior node that are terminal. In a subtree dominated by an interior node there will typically be (b-p)^h nonterminal nodes and p.(b-p)^(h-1) terminal nodes at the h'th generation after the subtree's root. At a limited depth there is a finite number of nodes in a subtree, and furthermore search at non-leaf nodes may be locally curtailed in several circumstances: when any daughter, terminal or otherwise, of an OR node is found to have value True; and when any daughter of an AND node is found to have value False or value Unknown. If the k'th daughter of a node *N* is found to have such a value, then it is unnecessary to evaluate the remaining (b-k) subtrees each of typical size (b-p)^g + p.(b-p)^(g-1), where g is the number of generations (i.e., additional levels of tree) that might have been explored beneath *N*.

In progressive deepening, only nodes with value Unknown need to be regenerated and reevaluated, using a greater value of h. In the case of an OR node, all daughters must either have value False or Unknown. Only the Unknown daughters need to be regenerated and their subtrees searched again. If any one of them is found to be True, the others may be ignored forever after. In the case of an AND node, all daughters must either have the value True or Unknown. Where the node has previously been searched, h>0, one daughter will be Unknown because a previous search was too shallow to reveal its proper value, and there may or may not be other daughters whose value is Unknown because they were not searched at all due to curtailment arising from that first Unknown.

When the deepened search of the first Unknown daughter of an AND node is performed, it has the potential to reveal one of the values True or False, or to leave it still Unknown. In the event of a False or Unknown value, remaining daughters need not be searched. But when the hitherto Unknown daughter is found to have value True, it is necessary to perform search on the remaining daughters also, until one of three eventualities comes to pass: either some remaining daughter is found to have value False, in which case remaining daughters may be permanently ignored and the parent node assigned value False; or some daughter is found to have value Unknown, in which case remaining daughters may be temporarily ignored and the parent node assigned value Unknown; or no daughters remain and the value True can be attributed to the AND node parent.

Discovering by search that a hitherto Unknown node is in fact True may trigger a cascade of evaluations of its sister nodes and their dominated subtrees. Where the search is allowed to proceed to depth h from the AND node parent, and it is the k'th daughter that has upgraded from Unknown to True, there will be unleashed a burst of activity with potentially (b-k) evaluations of subtrees of height (h-1). In Figure 3, for example, the search of node D (the second daughter of Node B) has just gone deep enough to reveal a 'True' value for it. Until then, there is no point[3] in evaluating the remaining daughters. But when this 'True' value is established, there is no longer any justification for pruning the search of B's next daughter, and it (and indeed possibly *all* remaining daughters) may need to have their subtrees expanded and evaluated.

§3. Recurrent Progressive Deepening search of the logjam

These subtree evaluations may be computationally expensive. This has the undesirable consequence that the time-control benefit of progressive deepening may be compromised. Time control is typically exercised by extrapolating from the pattern

[3] Evaluation of the remaining daughters does have the potential to reveal a 'False' value. However, if move ordering is believed to be good, the earlier daughters are the ones most likely to provide false values, that is, refutations by the opponent of a line of play.

of time usage and the observed branching factor on the levels of search conducted so far, to determine whether there is sufficient time remaining to search at least one level deeper. However it is possible that the *burst of activity* following the upgrading of just one interior node involves more, perhaps far more, computational effort than anticipated, fatally invalidating the prediction of the sufficiency of time available. For example, in figure 3, the cost of expanding two nodes at depth 4 (shown on grey background) and evaluating a condition at two nodes at depth 5 (on grey) provides no guide whatsoever as to the potential cost of expanding and evaluating to depth 5 the subtrees dominated by the third fourth and fifth daughters of Node B.

A second concern is that deep but futile searches may be performed that could and should be pruned. An interior AND node may be reckoned False if just one daughter is False, similarly an OR node True with just one daughter True. While a shallow search of some later daughter might be sufficient to provide such a result (if the move ordering is not optimal), this will not be known in time for the earlier daughters' subtree searches. But when a depth-first search is performed to a depth limit exceeding 1, time will be spent on fully exploring the first branch before the later branches are considered at all, and opportunities for pruning will be squandered.

The situation overall may be likened to a blockage that is suddenly cleared. A solution to the difficulties is simple and obvious. The tap should be opened slowly; the searches of the subtrees now eligible to be searched should be conducted in a progressive-deepening fashion, rather than being allowed to proceed immediately to the full depth (h-1). The *burst of activity* should be undertaken but in a controlled and moderated fashion.

By using progressive deepening in this way, time control can be exercised at each successively deeper level of the unexplored part of the subtree, preventing an irrevocable commitment to an arbitrarily large search. Pruning of futile branches may occur naturally also. Furthermore, in the event that move ordering at the parent node proved to be suboptimal, shorter refutation sequences are likely to be found sooner.

The traditional use of progressive deepening may be characterized as "First Order Progressive Deepening" (FOPD): the tree is searched to a succession of depths, with subtree searches being curtailed when demonstrably unnecessary. The term "Second Order Progressive Deepening" might be applied to the algorithm suggested above, where upgrading one daughter of an AND node exposes the forest of remaining daughters to FOPD. Clearly however this can be and should be generalized. With mere Second Order Progressive Deepening, the same problems could arise in the search of hitherto unexplored subtrees as arise in the FOPD case. Figure 4 shows an example of the sort of subtree that might be dominated by Node B in Figure 3, suppressing the detail of the first two daughters already shown in Figure 3, and also of one interior node whose value is still unknown after search to the depth of nodes D and E. FOPD would immediately commit to searching the entire subtrees shown, which would fail to exploit the opportunities for pruning afforded by the T values shown on a grey background, and might also overrun time controls. Second Order Progressive Deepening would search B's third subtree to depth 1, then 2, then 3, and

would run into exactly the same issues at Node E and its siblings such as Node F as have been noted for Node B and its siblings. While it may not be immediately apparent with the low narrow trees it is practical to show in this paper, it should be clear that these problems could be serious in the deeper bushier trees that game programs typically work with.

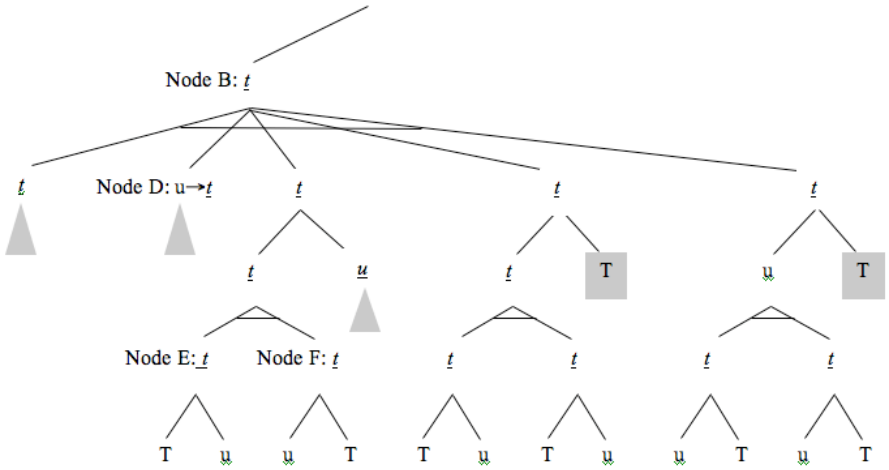


Figure 4
Second Order Progressive Deepening is not enough

"Recurrent Progressive Deepening" consists of searching a forest of trees (initially the trivial one-tree forest) to a succession of depths, curtailing tree and subtree searches where they are unnecessary. When an earlier shallower search on one tree had produced an Unknown result that curtailed search of its siblings (which constitute a forest), and a deeper search now produces an upgraded result that requires search of that forest, it too is searched using Recurrent Progressive Deepening[4] (RPD).

§4. Refinements: Transposition Tables and Deepening Schedules

Transposition tables (TTs) are well known to offer enormous speedups in game tree searches [6]. Their primary purpose is to store results of searches on positions, so that if a position can be reached by several different sequence of moves (a 'transposition') the subtree of moves and countermoves arising from it need not be searched more than once. Typically TTs are organized as hash tables, using numeric keys generated from positions by a fast method such as that of [7]. Numerous principled policies exist for handling the problem of collision, where several positions happen to be

[4] The term "Recursive Iterative Deepening" has been used [8] to denote a kind of proof-number search [5] in which progressive deepening (in a sense where 'depth' is taken to mean a limiting threshold on proof and disproof numbers) is performed not only at the root node, but also at every interior OR node.

represented by the same numeric key. One quite effective optimistic policy is to use large tables (a quarter of a million entries or more) and large keys (64 bits typically), and rely on the fact that collisions are then very rare in practice.

Transposition tables may store additional information serving various secondary purposes, such the depth of search that was conducted from a position to establish the main result, the best known move from a position, and indeed others. In Progressive Deepening, the depth of search that has yielded an unknown result can be compared to the current mandated depth of search: if the previous search was shallower, then a deeper search must be performed and the TT entry updated. Naturally a search to the mandated depth must also be performed if no record of the position exists, and a TT entry created. At the frontier of search most positions will not have been visited before, so TT entries will have to be created, evaluated as True or False or Unknown-with-zero-search as the case may be.

On occasions when the search horizon deepens beyond a node and that node still has an Unknown result, it will be visited again on a later iteration unless (a) search is terminated due to timeout or (b) a definitive result is found elsewhere in the tree. Then the TT may be used in a variety of ways. It may for instance provide the first of the daughters that provided an Unknown result; in the event that the node is an AND node and that wayward daughter is still unknown at the next depth of search, the cost of move generation to recover other daughter nodes is avoided. (Likewise, if an OR Node has only one Unknown daughter, and the TT records that fact, the cost of move generation may be avoided.) It may alternatively provide a list of all the moves leading to Unknown results, but the cost of this often outweighs the benefits. Or thirdly, move generation may produce the moves, the lookup keys for the resulting positions may be generated without necessarily first producing the positions themselves, and the TT may be consulted with the keys for the daughter nodes. This may well be a poor trade of computation time for a reduction in the size of TT entries unless move generation, key generation and lookup are all extremely quick.

Progressive deepening is conducted according to some deepening schedule, which specifies how many more levels are searched on one iteration than on its predecessor. Normally this schedule is not explicitly considered, and the frontier simply advanced by one generation (one ply). In goal-oriented searches of AND/OR game trees however this may not be the only sensible choice. It may be the case that only moves made by the achiever (opponent) have the potential to directly yield a True (False) result. In the Chess pawn-promotion example of figure 1 earlier, this is not quite so[5]. But in the Go stone-connection example, only achiever moves can achieve the goal, only opponent moves can frustrate it. In such cases, rather than evaluating every leaf node hoping to determine its value as True or False, it may be preferable to test AND nodes only to see if the goal has been achieved, and test OR nodes only to see if it has been frustrated. This is cheaper particularly when the logic of the tests is nontrivial.

[5] The fly in the ointment is that checkmate or stalemate by the achiever, or promotion of the pawn to some piece other than Queen, should cause False results.

This asymmetry may be exploited in a deepening schedule which advances the horizon of search by a double ply (moves plus responses). Such searches will always reach maximum depth at nodes of the same kind. If maximum depth is always reached at AND nodes, say, then the logic for testing achievement of the goal need only be applied at the limiting depth of search (depth remaining = 0), and the logic for testing frustration need only be applied at the level immediately before (depth remaining = 1). This optimization depends however on the assumption that in an progressive-deepening search the results for more distant ancestor nodes have been first stored in a transposition table, and then been retrieved and found to be Unknown.

If 'recurrent progressive deepening' has been used this assumption will hold. In contrast, the burst of activity characteristic of FOPD may result in many levels of a subtree being visited for the first time. This may result in moves and countermoves being tried well after the goal has been definitively achieved or definitively frustrated, which would be wasteful and could even lead to misleading search results.

It is sensible to consider double ply deepening schedules only when the branching factor is fairly low, since the number of new distinct nodes at level d+2 is greater than the number at level d by a factor of order b^2 (transposition tables may reduce the number below a simple factor of exactly b^2). If the branching factor is very low (as in the case of ladder capture in Go where it is usually little more than 1.0) then it may even be appropriate to consider many-ply deepening schedules. In cases of very low branching factor, the naïve intuition that progressive deepening is wasteful is quite correct: a considerable fraction of the total effort is expended on re-generating the upper levels of a tree. Where many-ply deepening schedules are in use, with limit = 2.n plies, the test for goal achievement should be done at every remaining-depth level d where d is even and d < limit, and the test for goal frustration when d is odd and d < limit. Further, when a *burst of activity* is being moderated through recurrent progressive deepening, the depth limits of successive searches should be synchronized with those that would have pertained if the subtree searches had never been curtailed.

§5. Conclusions

It has been argued here that there is a problem with First Order Progressive Deepening of a depth-first search of an AND/OR game tree. Specifically, when a relatively shallow search of a branch of any AND node has yielded an inconclusive result, the search of later branches can profitably be curtailed. If some subsequent deeper search establishes that branch as True, then the searches of remaining branches should go ahead. However, searching them may lead to a sudden burst of activity as they are each searched to possibly great depth. This compromises one important advantage of progressive deepening, namely its time control affordance; it also may result in unnecessary effort expended in that search, since sufficient proofs or refutations that might be discovered in later branches at shallow levels must wait to be discovered after deep searches of early branches.

The simple technique of recurrent progressive deepening is proposed to overcome these difficulties. It consists of managing the burst of activity, with searches of the remaining branches being performed according to the same deepening schedule as if they had never been curtailed at all, including the use of recurrent progressive deepening if similar situations should arise during those searches. Future work is required in order to determine quantitatively how much calculation is thereby saved, and how frequently in practice an unmanaged burst of activity invalidates the time control decision making that is part of the attraction of progressive deepening.

The argument has been presented in the context of searches of AND/OR trees, where the issues are particularly clear cut. However, any depth-first search technique that uses pruning of any kind and also uses progressive deepening will be susceptible to the same problems and the same remedy. When an alpha-beta search determines a backed up value for a successor node which exceeds its local beta parameter, the search of other successor nodes is curtailed. If the same node is revisited, and a deeper search reveals a backed up value not exceeding the local beta (whether changed or unchanged is irrelevant), then the subtrees of the other successor nodes will be searched. This search, if carried out to the currently permitted search depth, will manifest exactly the same problems as are seen in AND/OR trees. The remedy of recurrent progressive deepening should be widely applied.

References

1. Scott, J.J. 1969. A chess-playing program. In Machine Intelligence 4, B. Melzer and D. Michie (eds), pp255-265. Edinburgh University Press., Edinburgh, Scotland.

2. Reinefeld, A. and Marsland, T.A. 1994. Enhanced Iterative-Deepening Search. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 16 no 7, pp701-710.

3. Knuth, D.E. and Moore, R.W. 1975. An analysis of Alpha-Beta Pruning. Artificial Intelligence 6, pp293-326.

4. Fotland,D. Computer Go Design Issues. Retrieved 10 August 2007. http://www.inventivity.com/OpenGo/Papers/Zipped/KnowldgeRep-Fotland.txt.zip

5. Allis, V. Proof-Number Search. Artificial Intelligence 66 no 1, pp91-124, 1994.

6. Marsland, T.A. The Anatomy of Chess Programs. Retrieved 10 August 2007. http://www.cs.ualberta.ca/~tony/ICCA/anatomy.html

7. Zobrist, A.L. 1970. A new hashing method with applications for game playing. Tech. Report 88, U. Wisconsin. Reprinted in 1990 in International Computer Chess Association Journal, vol 13 no 2, pp69-73.

8. Seo, M., Iida, H., and Uiterwijk, J. 2001. The PN*-search algorithm: Application to tsume-shogi. Artificial Intelligence 129 pp253-277.

Reasoning about Durative Action

Karl Devooght[1,2] and Marc Guyomard[2]

[1] France Tlcom R&D
2, avenue Pierre Marzin, 22300 Lannion
[2] LLI-IRISA, 6, rue de Kerampont, 22300 Lannion
karl.devooght@orange-ftgroup.com

Abstract. Traditionally, existing logics for agency, particularly modal logics, are based on actions that are atomic. Atomic actions are actions which are executed without any account of possible interferences from the environment and have no duration. In this paper, we aim at providing logical supports for reasoning with another kind of actions: *durative* action. A durative action is basically an action which takes time to be executed. In this view, we propose to extend the *Propositional Dynamic logic* (PDL) in order to reason with both atomic and durative actions.

1 Introduction

Traditionally, existing logics for agency, particularly modal logics, are based on actions that are atomic. *Atomic* actions are actions which are constrained by the two following hypotheses:

- they are executed without taking into account of possible interferences from the environment,
- they have no duration.

However, several real-time applications domains, namely in Artificial Intelligence (e.g. agent programming) and in Databases (e.g. transactions management) requires logical support for *durative* action. Durative actions are actions which take time to be executed.

Let us give a concrete example. Consider an autonomous robot which has to do different mechanical tasks. Among these tasks, it needs to rotate its metallic arm by 90 degrees. Suppose that executing the action of rotating takes ten seconds. In a real-time environment, it is quite intuitive there may be an event that hampers the execution of this action. For example, some events may arrest the agent's arm during its movement. It would be useful it could abort its action in these conditions.

This simple example suggests that modeling the action of rotating as an atomic action is improper. That is, such an action violates both hypotheses constraining atomic action: it takes ten seconds and some interference may appear

during its execution. To our knowledge, the common given response[3] is to break such an action up in atomic actions. More precisely, a durative action is seen as a sequence of atomic actions. Thus, if one wants to express some conditions which occur during the execution, one must consider them as pre-conditions of a compounding atomic action. But what level of refinement does it require in order for an agent to be sufficiently reactive during an execution? More important, let's suppose one finds such a refinement, why does one complicate an agent model with additional actions to represent ultimately a basic mechanical action as the action of rotating?

It appears that an agent is required to deal with the *continuity* of certain actions. In other words, we need a richer logical system which allows an agent to reason with durative actions. For atomic actions, the *Propositional Dynamic Logic* (PDL) [4] is a natural candidate. In this paper, we propose to extend this logic for durative action and to augment it with two new modalities such that one may reason with both atomic and durative action. The first modality expresses *what is true whenever a durative action is being executed*. The second one says that *after each ongoing execution of durative action, something is true*. We argue that both modalities point out the striking features about the phenomenology of durative actions while they present themselves as natural extensions of existing PDL operators for atomic actions.

The paper is structured as follows. Section 2 informally introduces the notion of durative actions. A formal account of durative actions is given in section 3 as part of a logic extending PDL. The proposed logic is sound. We illustrate it with our introductory example. In section 4, we compare our approach with some recent works on durative actions.

2 Reasoning about Durative Actions

In agency analysis, two classes of examples are used concerning actions [6]. On the one hand, consider the following examples (inspired from Davidson's work [2]):

- The doctor removed the patient's appendix.
- John will fly to the north pole.

On the other hand, in *intention* [1], Anscombe mostly uses similar examples of the following kind:

- Mary is peeling an apple.
- When Mary is peeling an apple, she may cut her finger with a knife.

In Davidson-type examples, the past and future tenses are used. In this case, action is considered in the perfective aspect (from a point of view after the action is/will be finished). Anscombe, however, uses the present tense. This involves considering action in the imperfective aspect (from a point of view

[3] Some exceptions will be discussed in the Related works section.

while the action is occurring). Analytical investigations of agency have mostly been concerned with Davidson's examples. In fact, most of action logics follow in this manner. In these approaches, logical supports relate actions to both the states of affairs *before* and *after* their execution. Thus, a natural hypothesis is to see actions as atomics. For instance, consider apple peeling. In this case, the situation is simple enough. First, the apple is not peeled, then the apple is peeled. Although this account of action is efficient in many cases, Anscombe-type examples of durative actions need to be covered for some real-time agent-based applications (as one described in section 1).

While an agent is acting, the phenomenology of durative actions is different from atomic actions. A durative action such as apple peeling may be described as follows:

- Some time in the past, the agent decided and then started to peel the apple.
- He was peeling up to now.
- By default, he tends to peel until he reaches a state of affairs where his apple is peeled.

In accordance with this description, we propose to characterize informally an execution of durative action as follows:

1. A particular atomic action causes the beginning of action execution.
2. There is some course of actions during the ongoing execution.
3. After which, one is in a state of affairs from which a particular atomic action involves the end of action execution.

As for atomic actions, an execution of a durative action is bounded i.e. by the states before and after its execution. In our approach, the fundamental difference between them is that the spent time or more precisely the number of actions occurred during the execution of a durative action is neither nil nor fixed.

Besides, in regard of this characterization, crucial aspects for durative action have to be highlighted. First, an agent needs to be aware about some knowledge and events which may be stated during an action execution. Intuitively, getting this information is important for the agent since it may influence his future choices. Knowing what may happen during an action execution may lead an agent deciding whether to execute it or not. For example, suppose that an agent knows that while he is rotating his metallic arm, an event may trouble its execution. Then, he may decide not to do it in a situation where this event has a strong chance of appearing. Second, while the agent is peeling an apple for instance, this does not strictly imply that this apple will actually be peeled. There are various reasons which force the agent to abort the execution of his action. For instance, cutting his finger with a knife may be a good example. In linguistics, one calls it the *imperfective paradox*. However, the default agent choices during the action execution are made in order to reach a state of affairs where the apple is peeled. It involves that to be performing is goal-directed in some sense.

Modeling logically an account of durative action requires to explicitly characterize these aspects. We argue that reasoning with durative actions needs to

support at least two kinds of modalities. The first one denotes *what is true whenever an agent is executing an action*. This modality allows an agent to reason about what happens during the execution of some action (even if he is not actually executing it). Thus, an agent may prevent interferences from his environment during his ongoing action. For example, cutting his finger when he is peeling an apple may be envisioned. In addition, it is evident that the first hypothesis on atomic actions is not valid for durative actions with the existence of such a modality. The second modality says that *after each ongoing execution of a durative action, something is true*. This states that a durative action takes time. This invalidates the second hypothesis on atomic actions for durative action. Moreover, this modality underlies there are some expected future states towards which an agent is acting. For example, this modality allows us to express that an agent is peeling an apple in such a way that the apple is peeled afterwards.

3 Extending PDL with durative action

In this section, we propose a logic for reasoning about durative actions. More precisely, this logic extends the Propositional Dynamic Logic (PDL) in order to reason about both atomic and durative actions.

3.1 Syntax

Let us start with the syntax of our logical language $\mathcal{L}$ which is basically a restricted PDL language augmenting by two new modalities $\{a_d\}$ and $\Box a_d \Box$. $\{a_d\}\phi$ means that *ϕ is true whenever the durative action a is being executed*. $\Box a_d \Box \phi$ means that *after each ongoing execution of durative action a_d, ϕ is true*. The language $\mathcal{L}$ contains a set of propositional variables $\mathcal{P}$, a set of atomic actions $\mathcal{A}_a$ and a set of durative actions $\mathcal{A}_d$. We denote $\mathcal{A}$ the set of all actions i.e. the union of $\mathcal{A}_a$ and $\mathcal{A}_d$. Formulae ϕ and actions a of $\mathcal{L}$ are then defined as follows:

$$\phi ::= \top \mid p \mid \phi \vee \phi \mid \neg\phi \mid [\alpha]\phi \mid \{a_d\}\phi \mid \Box a_d \Box \phi$$
$$\alpha ::= a \mid a_d \mid \alpha; \alpha$$

where $p \in \mathcal{P}$, $a \in \mathcal{A}_a$ and $a_d \in \mathcal{A}_d$. $\top$ denotes the logical truth.

Other formula operators can be introduced as abbreviations: $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi$, $\phi \Rightarrow \psi := \neg\phi \vee \psi$, $< a > \phi := \neg[a]\neg\phi$ and $\bot := \neg\top$. The formula $< a > \top$ is traditionally read as *an execution of action a is possible*. In addition, we set $\diamond a_d \diamond \phi := \Box a_d \Box \phi$ which means that *there is an ongoing execution of durative action a_d after what ϕ is true*. Thus, $\diamond a_d \diamond \top$ is understood as *action a_d is being executed*. On the contrary, $\Box a_d \Box \bot$ means that *action a_d is not being executed*

Actions can be primitive i.e. atomic or durative, or complex i.e. a sequence of atomic or durative actions. In addition, for each durative action $a_d \in \mathcal{A}_d$, there exists three related atomic actions $b(a_d)$, $e(a_d)$ and $ab(a_d)$ in $\mathcal{A}_a$. Action $b(a_d)$ (respectively $e(a_d)$) begins (respectively ends) the execution of action a_d. Action $ab(a_d)$ aborts its execution.

3.2 Semantics

Traditionally, PDL formulas are interpreted over Kripke models [5]. We follow this way by considering Kripke models $\mathcal{M} = (\mathcal{S},\ \stackrel{a}{\rightarrow} \mid a \in \mathcal{A},\ \mathcal{D}_{a_d} \mid a_d \in \mathcal{A}_d,\ \mathcal{E}_{a_d} \mid a_d \in \mathcal{A}_d,\ \mathcal{I}_p \mid p \in \mathcal{P})$. These models consist of (1) a set of state of affairs $\mathcal{S}$, (2) $\stackrel{a}{\rightarrow}$ for each action $a \in \mathcal{A}$ is a binary relation from states before the execution of a to those afterwards, (3) $\mathcal{D}_{a_d}$ for each $a_d \in \mathcal{A}_d$ is a binary accessibility relation to states where the agent is executing a_d,(4) $\mathcal{E}_{a_d}$ for each $a_d \in \mathcal{A}_d$ is a binary accessibility relation from states where a_d is being executed to states after its execution, and (5) $\mathcal{I}_p$:$2^{\mathcal{S}}$ an interpretation of the proposition p in $\mathcal{M}$.

The semantics of a sequence of actions and a durative action can be explained by induction on actions. Let s,$t \in \mathcal{S}$.

$$s \stackrel{a1;a2}{\rightarrow} t \text{ iff } \exists u \in \mathcal{S} \text{ s.t. } s \stackrel{a1}{\rightarrow} u \text{ and } u \stackrel{a2}{\rightarrow} t$$

$$s \stackrel{a_d}{\rightarrow} t \text{ iff } \exists u, v \in \mathcal{S} \text{ s.t. } s \stackrel{b(a_d)}{\rightarrow} u,\ \mathcal{E}_{a_d}(u,t), \text{ and } v \stackrel{e(a_d)}{\rightarrow} t$$

This semantics meets our point about durative action discussed in section two. An execution of a durative action a_d is characterized by the execution of action $b(a_c)$ beginning after which a_d is being executed. The ongoing execution of a_d is then directed towards a state to which it has been ended by action $e(a_d)$. In particular, we do not require to explicitly specify the course of actions during an execution of durative action.

Now, we may define the satisfiability relation $\models$. Let ϕ be a well-formed formula (wff) of $\mathcal{L}$, $\mathcal{M}$ a model, and s a state of $\mathcal{S}$. $\mathcal{M}$,$s \models \phi$ stands for ϕ is *satisfied* (i.e. is *true*) in the state s of $\mathcal{M}$. ϕ is *valid* iff $\mathcal{M}$,$s \models \phi$ for all pairs ($\mathcal{M}$,s). The satisfiability relation $\models$ is thus recursively defined as follows:

- $\mathcal{M}$,$s \models \top$;
- $\mathcal{M}$,$s \models$ p iff $s \in \mathcal{I}_p$;
- $\mathcal{M}$,$s \models \neg\phi$ iff $\mathcal{M}$,$s \not\models \phi$;
- $\mathcal{M}$,$s \models \phi \vee \psi$ iff $\mathcal{M}$,$s \models \phi$ or $\mathcal{M}$,$s \models \psi$;
- $\mathcal{M}$,$s \models [a]\phi$ iff $\forall t \in \mathcal{S}$ s.t. $s \stackrel{a}{\rightarrow} t$, $\mathcal{M}$,$t \models \phi$;
- $\mathcal{M}$,$s \models \{a_d\}\phi$ iff $\forall t \in \mathcal{S}$ s.t. $\mathcal{D}_{a_d}(s,t)$, $\mathcal{M}$,$t \models \phi$
- $\mathcal{M}$,$s \models \Box a_d \Box \phi$ iff $\forall t \in \mathcal{S}$ s.t. $\mathcal{E}_{a_d}(s,t)$, $\mathcal{M}$,$t \models \phi$

Note that the traditional dynamic operator $[a]$ can be applied on both atomic and durative actions.

In terms of semantic constraints, we impose each relation $\mathcal{D}_{a_d}$ to be:

- serial: $\forall s \in \mathcal{S}$, $\exists t \in \mathcal{S}$ s.t. $\mathcal{D}_{a_d}(s,t)$,
- transitive: $\forall s, t, u \in \mathcal{S}$ s.t. $\mathcal{D}_{a_d}(s,t)$ and $\mathcal{D}_{a_d}(t,u)$, $\mathcal{D}_{a_d}(s,u)$,
- euclidian:$\forall s, t, u \in \mathcal{S}$ s.t. $\mathcal{D}_{a_d}(s,t)$ and $\mathcal{D}_{a_d}(s,u)$, $\mathcal{D}_{a_d}(t,u)$.

But no particular constraint is required on relations $\mathcal{E}_{a_d}$.

3.3 Axiom system

We propose an axiom system for our logic. In regard of semantic constraints, the modal operator $\{a_d\}\phi$ is a $KD45$-operator while the operator $\Box a_d \Box \phi$ is a K-operator. Existing PDL axioms for operator $[a]$ are still valid in our approach. The rules of our logic are *Modus Ponens* (MP), and *Necessitation* rules for each modal operators.

$$\text{If } \vdash \phi \text{ then } \vdash [a]\phi$$
$$\text{If } \vdash \phi \text{ then } \vdash \{a\}\phi$$
$$\text{If } \vdash \phi \text{ then } \vdash \Box a_d \Box \phi$$
$$\text{If } \vdash \phi \text{ and } \vdash \phi \rightarrow \psi \text{ then } \vdash \psi$$

All axioms of propositional logic
$$\vdash [a]\phi \wedge [a](\phi \rightarrow \psi) \rightarrow [a]\psi$$
$$\vdash [a1; a2]\phi \rightarrow [a1][a2]\phi$$
$$\vdash \{a_d\}\phi \wedge \{a_d\}(\phi \rightarrow \psi) \rightarrow \{a_d\}\psi$$
$$\vdash \{a_d\}\phi \rightarrow \neg\{a_d\}\neg\phi$$
$$\vdash \{a_d\}\phi \rightarrow \{a_d\}\{a_d\}\phi$$
$$\vdash \neg\{a\}\phi \rightarrow \{a\}\neg\{a\}\phi$$
$$\vdash \Box a_d \Box \phi \wedge \Box a_d \Box(\phi \rightarrow \psi) \rightarrow \Box a_d \Box \psi$$

In regards of the definition of $\stackrel{a_d}{\rightarrow}$, this kernel of axioms do not suffice to get a more precise account of durative action. What follows are considerations of further axioms and their semantic counterparts.

First, there is a strong link between operators $\Box a_d \Box$ and $[a]$ (where a is a durative action). Suppose that after each execution of a durative action ϕ is true. What we may say is that ϕ is true after each *ongoing* execution of this action as well. We have the following axiom and its semantic counterpart:

$$\vdash [a_d]\phi \rightarrow \Box a_d \Box \phi$$
$$\forall s,t \in \mathcal{S} \text{ s.t. } \mathcal{E}_{a_d}(s,t), \exists s' \in \mathcal{S} \text{ s.t. } s' \stackrel{a_d}{\rightarrow} t$$

Second, it sounds like a tautology that a durative action is being executed whenever it is being executed. We obtain:

$$\vdash \{a_d\} \diamond a_d \diamond \top$$
$$\forall s,t \in \mathcal{S} \text{ s.t. } \mathcal{D}_{a_d}(s,t), \exists t' \in \mathcal{S} \text{ s.t. } \mathcal{E}_{a_d}(t,t')$$

Third, since we work with action-types and not action tokens, overlapped executions of the same action need to be avoided. To do so, we require that after each execution of durative action, it is not being executed:

$$\vdash [a_d]\Box a_d \Box \bot$$
$$\forall s,t \in \mathcal{S} \text{ s.t. } s \stackrel{a_d}{\rightarrow} t, \nexists t' \in \mathcal{S} \text{ s.t. } \mathcal{E}_{a_d}(t,t')$$

From this axiom and axiom $[a_d]\phi \rightarrow \Box a_d \Box \phi$, we may easily infer that being acting is always expected to reach a state where one is not acting i.e. $\Box a_d \Box \Box a_d \Box \bot$.

Fourth, the particular atomic actions $b(a_d)$, $e(a_d)$ and $ab(a_d)$ have systematic effects related to the fact of being acting or not. We may sum up it as follows: (1) if one has just started executing a durative action, one is acting and (2) if one has just ended or finished, one is not acting. In addition, if an execution of a durative action is possible, that implies that it is possible to launch this execution. We have the following axioms:

$\vdash < a_d > \top \rightarrow < b(a_d) > \top \wedge [b(a_d)] \diamond a_d \diamond \top$

This axiom comes from the definition of $\overset{a_d}{\rightarrow}$

$\vdash < e(a_d) > \top \rightarrow \diamond a_d \diamond \top \wedge [e(a_d)] \Box a_d \Box \bot$

$\forall s,t \in \mathcal{S}$ s.t. $s \overset{e(a_d)}{\rightarrow} t$, $\exists t' \in \mathcal{S}$ s.t. $\mathcal{E}_{a_d}(s,t')$ and $\nexists t" \in \mathcal{S}$ s.t. $\mathcal{E}_{a_d}(t,t")$

$\vdash < ab(a_d) > \top \rightarrow \diamond a_d \diamond \top \wedge [ab(a_d)] \Box a_d \Box \bot$

$\forall s,t \in \mathcal{S}$ s.t. $s \overset{ab(a_d)}{\rightarrow} t$, $\exists t' \in \mathcal{S}$ s.t. $\mathcal{E}_{a_d}(s,t')$ and $\nexists t" \in \mathcal{S}$ s.t. $\mathcal{E}_{a_d}(t,t")$

Finally, operators $\{a_d\}$ and $\Box a_d \Box$ are naturally related. If one is acting and some ϕ is true whenever one is acting, ϕ is then true. The following axiom is considered:

$$\vdash \diamond a_d \diamond \top \wedge \{a_d\}\phi \rightarrow \phi$$
$$\forall s,t \in \mathcal{S} \text{ s.t. } \mathcal{E}_{a_d}(s,t), \mathcal{D}_{a_d}(s,s)$$

So, this means that operator $\{a_d\}$ is also used in order to contextually infer some facts i.e. the context of being acting.

Considering all these rules and axioms, soundness for this logic can be easily proven by induction on the size of formula and by using all the semantical counterparts. A completeness result is planned for further works.

3.4 Example

We illustrate the logic with our introductory example. Consider an autonomous robot whose task is to rotate its metallic arm horizontally by 90 degrees. Suppose that this task takes ten seconds. While he is acting, some event may cause its arm to be arrested. Our logic allows to express different scenarios involving durative actions.

Consider *turning* and *removing* as durative actions meaning the actions of *rotating its arm* and *removing what arrests the robot's arms during its rotation.* Consider also the propositional constants *arrested* meaning that *the robot arm is arrested*, and *turned* meaning that *the robot arm has been rotated horizontally at 45 degrees.*

Let us imagine some situations that our logic can express. First of all, there may exist some conditions which allows the robot to successfully end his action

of rotating. Suppose that this is the case whenever the propositional constant *turned* is true. Formally, we described it as follows:

$$\{turning\}(turned \rightarrow < e(turning) > \top)$$

More important are the cases when the robot's arm is arrested while he is rotating it. For example, a particular action may imply *arrested* to be true. One may suppose that gives the agent a possibility to abort his action:

$$\{turning\}(arrested \rightarrow < ab(turning) > \top)$$

This case is pretty optimistic since, for instance, this action may imply a strong consequence: the agent is not rotating anymore. Let us call this action *arresting*. We describe such a scenario by:

$$\{turning\}(< arresting > \top \wedge [arresting]\Box turning\Box\bot)$$

On the contrary, there may be a back-up action which enables an agent to remove what arrests its arm. We called this action *removing*. So, during the agent's arm rotation, the action *removing* is possible and if the arm is arrested, the agent may remove what is wrong:

$$\{turning\}(< removing > \top \wedge (arrested \\ \rightarrow [removing]\neg arrested \wedge \Box turning\Box\top))$$

Note that this scenario shows the possibility of concurrent executions of two durative actions i.e. $\diamond turning \diamond \top \wedge \diamond removing \diamond \top$.

Suppose now that one does not want the robot to rotate its arm if there is a possibility that it is arrested, and in this case, to cause the impossibility to execute the action of rotating. Such a situation may definitely be expressed with our logic as follows:

$$(\{turning\} < arresting > \top) \rightarrow [turning]\bot$$

This is a typical example of the usefulness to know *before* what may happen *during* the execution.

4 Related works

The phenomenology of durative actions is the object of studies in different fields like in philosophy e.g. Searle's work [7] and in linguistics e.g. Vendler's work [12]. In artificial intelligence, namely in agent modeling, there are two kinds of approaches depending on the representation of time. In the first approach, time is *continuous* and *explicitly represented* in the semantical model [3,8,9]. So, all actions take time and formulas are evaluated at time points and may be interpreted over time intervals. In the second approach, time is *discrete* and generally *implicitly represented*. Our work follows this way and intends to show

that reasoning about durative actions does not especially require to consider neither syntactically nor semantically explicit duration of an action.

Since very recently, one finds formal accounts of logical modalities related to durative action. In particular, Müller's [**?**] and Troquard's [10,11] works have investigated this research avenue.

As part of a STIT-frameworks i.e.*See To It That*, Müller introduces a modal operator *istit*. The operator *istit* means that *an agent is seeing to it that*. The semantics of *istit* is based on the notion of strategy. A *strategy* is basically a subset of agent's default choices leading him towards particular agent's futures. Then, for a particular propositional constant ϕ, an agent is seeing to it that ϕ is true if and only if (1) ϕ is true in all the future states of affairs reachable by following the agent's strategy, and (2) there exists a future state of affairs in which ϕ is not satisfied. The second condition of the definition is called the *counter* condition. Müller's approach is quite different from ours. First, in STIT-framework, action is implicitly represented. So, an example like "Mary is peeling an apple" cannot be logically expressed (but rather "Mary is seeing to it that the apple is peeled"). Second, we do not impose a counter condition since our operators are assigned to a more generic interpretation i.e. *being executing*. Nevertheless, it is possible to extend our logic with an *istit*-like operator defined as follows: an agent is seeing to it *with an action* a_d that ϕ is true if and only if (1) he is executing a_d after what ϕ is true, and (2) there is a possible action involving ϕ false e.g. aborting a_d.

Troquard's works go a step further. They combine STIT and PDL logic. They refine the operator $[\beta : a]\phi$ with the following meaning: an agent a starts performing the action β and ϕ is true after the execution of β. Similarly, our approach requires a durative action to be related to an atomic action beginning its execution. Then, Troquard characterizes the durative aspect of action with particular (atomic) actions, called *continuations of an action*. These actions are introduced as an artefact in order for an agent to control the durative execution of his action. Their possible execution also underlies that an agent is actually executing his action. So, if this action is not possible, that implies that an agent is not executing it anymore. We believe that it is unclear to what these actions relate to. First, if they correspond to concrete agent observations of the action execution, we argue that this characterization of durative action is too strong. Indeed, that underlies to break a durative action up as a sequence of atomic actions (see section 1). Second, if these actions allows an agent to know that he is acting, it is more intuitive and expressive to define a modal operator dedicated to this role (e.g., the operator $\diamond a_d \diamond \top$). Third, if they correspond to actions allowing an agent to end its execution, one requires only a particular action terminating it effectively (e.g, for a durative action a_d, an atomic action $e(a_d)$).

5 Conclusion

In this paper, we proposed a formal account of durative action. In particular, we introduced two modal operators as part of a proposed logic extending PDL.

The first operator says that something is true whenever an agent is executing a durative action. It enables us to mention facts appearing during the action execution. More precisely, interferences from the environment during the execution may be expressed. The second operator says that an agent is executing an action and something is true afterwards. It underlies that some actions may take time. Besides, it means that when an agent is acting, he tends to reach some particular futures states by default. But, as we showed, executions of durative action can be aborted intentionally with a particular action $ab(a_d)$ or not. Furthermore, we have shown that our logic enables a reasoning with both atomic and durative actions.

The area of perspectives is open. It would be interesting for example to study traditional complex actions as sequences of actions or actions-loop with regard to durative action. In the field of planning, operator $\Box a_d \Box$ can be used in order to keep a track of the ongoing agent plan. Suppose that the sequence of action $a1; a2$ is considered as a durative action. A formula like $\Box a1; a2 \Box$ involves the agent to be aware about the plan that he is carrying out. Furthermore, reasoning with concurrent actions is a subject of much interest in the scope of our logic.

References

1. G.E.M. Anscombe. *Intention.*. 2nd ed., Oxford, 1963.
2. D. Davidson. *The Logical Form of Action Sentences.* In his *Essays on Actions and Events*, pp 105–122,Oxford, 1967.
3. E. Davis. *Branching Continuous Time and the Semantics of Continuous Action.* In proceedings of *Artifical Intelligence Planning System*, pp 231–236, 1994.
4. D. Harel. *Dynamic logic.* MIT Press, 2000.
5. S.A. Kripke. *Semantical Considerations on Modal logic.* In *Acta Philosophica Fennica*, Vol. 24, pp 83–94, 1963.
6. T. Müller. *On the Formal Structure of Continuous Action.* In *Advances in Modal Logic*, King's College Publications, 2004.
7. J. Searle. *Rationality in Action.* MIT Press, Cambridge, MA, 2001.
8. M.P. Singh. *Formalizing Actions in Branching-Time: Model-Theoretic Considerations.* In proceedings of *2nd Workshop on Temporal Representation and Reasoning*, Melbourne Beach, Florida, USA, 1995.
9. M.P. Singh. *Towards a Model Theory of Actions: How Agents do it in Branching-Time.* In *Computational Intelligence*, 1998.
10. N. Troquard, R. Trypuz and L. Vieu. *Towards an ontology of agency and action: From STIT to OntoSTIT.* In proceedings of *International Conference on Formal Ontology in Informational Systems*, Baltimore, Maryland, USA, B.Bennett, C.Fellbaum (Eds), IOS Press, Frontiers of Artificial Intelligence and Applications, pp 179–190, 2006.
11. N. Troquard and L. Vieu. *Towards a Logic of Agency and Actions with Durations.* In proceedings of *European Conference on Artificial Intelligence*, Riva Del Garda, Italy, pp 775–776, IOS Press, 2006.
12. Z. Vendler. *Verbs and Time.* In *philosophical Review*, Vol. 56, pp 143–160, 1957.

A Combinational Creativity Approach to Composing Traditional Irish Reels

Nan Zheng[1] , Bryan Duggan[1]

[1] School of Computing, Dublin Institute of Technology, Kevin St., Dublin 8, Ireland
nan.zheng1@student.dit.ie, bryan.duggan@dit.ie

Abstract. In this paper we describe a system that uses a corpus of 864 traditional Irish reels as input into an algorithm that composes new tunes. The system performs a structural analysis of the tunes in the corpus and also counts n-gram note sequences in the tunes. It then recombines n-gram note sequences together in structures from the corpus to generate new tunes. We further present our evaluation of the generated tunes as performed by 29 domain experts.

Keywords: Irish traditional music, reel, algorithmic composition, stochastic sampling, n-grams.

1 Introduction

The most common forms of traditional dance music are *reels*, *double jigs* and *hornpipes*. Other tune types include *marches*, *set dances*, *polkas*, *mazurkas*, *slip jigs*, *single jigs and reels*, *flings*, *highlands*, *scottisches*, *barn dances*, *strathspeys* and *waltzes* [1]. These forms differ in time signature, tempo and structure. A reel is generally played at a lively tempo and is in 4/4 time (although played and transcribed as 8 quavers in a bar). Most tunes have a structure consisting of two "parts" (of length eight bars each) called the A part and B part. In a double reel, each part is played twice and then the entire structure is repeated up to three times. Typically within each part there is a certain amount of repetition, which usually occurs in half bar phrases.

[2] describes combinational creativity as "*novel combinations of old ideas*". It is clear from any analysis of a corpus of traditional music that there are many examples of combinational creativity. The tunes "The Bag of Spuds" and "Down the Broom" have very similar B parts for example while the tunes "Sleepy Maggie" and "Jenny's Chickens" share very similar A and B parts. In this paper we describe our work in developing a combinational creativity algorithm that can compose new reels by first analysing the structure of tunes from a corpus and then recombining n-grams of notes from the corpus to create new reels. Our system takes advantage of the fact that half bars in tunes from the corpus often occur several times in a tune to create structures that should sound correct. The fact that note sequences in the generated tunes come from n-grams of notes in the corpus should also mean that the generated melodies are plausible. Our system generates any number of new tunes. To test the system, we generated one hundred new tunes using our algorithm and selected nine at

random for evaluation by domain experts. We included one human composed tune for comparison and asked the experts to try and guess which tune was composed by a human.

Section 2 presents background information on the corpus that we used as input to our system and presents the ethnographic context of the corpus. Section 3 of this paper describes related work in the field of algorithmic composition focusing of systems that purport to create folk melodies. Section 4 of this paper describes our algorithm in detail. Section 5 of the paper presents an evaluation of the generated tunes performed by 29 domain experts, while section 6 presents conclusions and future work.

2 Background

Current estimates suggest there are at least seven thousand traditional tunes in existence [3]. One of the main reasons proposed for the existence of such a wide repertoire is the geographic isolation of rural Irish communities in the centuries preceding the twentieth century [4]. It is proposed that many isolated rural communities developed their own repertoire of tunes and that widespread knowledge of a common repertoire did not occur until the publication of catalogues of traditional tunes such as O'Neill's The Music of Ireland in 1903. In his seminal work, O'Neill collected 1850 tunes played by emigrant Irish musicians in Chicago at the time. Several of the tunes in the catalogue are variations of the same tune.

In 1991, the ABC music notation language was introduced by Chris Walshaw [5]. The format was designed primarily for folk and traditional tunes of Western European origin which can be written on one stave in standard classical notation [5]. ABC files are ASCII text files and so can be edited by any text editor, without the necessity for special software. Each file (known as a tune book) can contain multiple tunes. File sizes are typically measured in kilo-bytes and this facilitates easy transmission by electronic means.

Figure 1 is the tune "Contentment is Wealth" in the ABC format. Each tune consists of a header section and a tune body. The header section contains amongst other fields, the title, com-poser, source, tempo, key signature, geographical origin and transcriber [6]. As tunes can have several titles, the title field can be repeated for a given tune.

```
X:11
T:Contentment is Wealth
R:jig
M:6/8
K:Edor
GFG Eed|BAB EFG|FAF DdB|AFD D2f|gfe edB|BAB ~d3|BdB
DFA|GED E3:|
|:ede Beg|bge gfe|dcd Adf|afd fed|ede Beg|bge gfe|BdB
DFA|GED E3:|
```

Figure 1: The tune "Contentment is Wealth" in the ABC format.

The tune body contains the notation for the tune. The body encoding supports such features as ornaments, bar divisions, sharps, flats, naturals, repeated sections, key changes, guitar chords, lyrics and variations. Between 1997 and 2000, a group of musicians under the leadership of Dan Beimborn and John Chambers, undertook a grass roots project to transcribe three of O'Neill's books to electronic format using the ABC music notation language. As copyright had expired on O'Neill's original books, they made their work freely available on the internet [7].

Many of the tunes from O'Neill's books are played differently by musicians today, as is normal with a living tradition. Around the same period (the late 1990's) Henrik Norbeck collected nearly 2000 tunes in ABC format from various sessions and recordings. Again this collection was made freely available on the internet. This collection contains many modern settings of tunes from O'Neill's books [8] Our system uses a corpus of 864 reels in ABC format drawn from Henrik Norbeck's transcriptions.

3 Related Work

[9] describes the CONCERT system. CONCERT is first trained on melodies represented as a sequence of note pitch names, durations and chords. Various corpora were used to train CONCERT including sets of J. S. Bach pieces and traditional European folk melodies. Internally, CONCERT uses a recurrent network architecture that learns to behave as an autopredictor. A melody is presented to it, one note at a time, and its task at each point in time is to predict the next note in the melody.

The author reports that while the system performed well on simple, structured, artificial sequences, the architecture failed to capture global musical structure. He reports that few listeners were fooled into believing that the pieces had been composed by a human and describes the output of his model as "music only its mother could love".

In [10], the authors describe a novel system that uses a neural network trained on one thousand traditional Irish tunes (jigs, reels and slow airs) and that uses Irish rainfall data as input to generate new melodies. They first convert the training set to MIDI and truncate each of the tunes to be less than 128 note events. They used two back propagation neural networks, one for pitch and one for duration and trained the networks to recognise each of the 1000 tunes from the training set. They then forced the normalised rainfall data upon the inputs to the neural networks, which resulted in the networks producing output vectors of 128 note events. As they had data for one year, they generated 365 note sequences. They developed a windowing system that extracted sequences of the generated melodies in order and re-sequenced them to form a playable melody. The authors provide no validation of the quality of the generated melody, but the generated melody was played by the Irish Chamber Orchestra and was chosen to be one of the centrepieces in the Irish Pavilion at EXPO2000 in Hanover. To our ear, the melody sounds atonal and lacking in musical structure however and quite unlike the tunes from the corpus.

In [11] the authors describe a system that identifies long timescale musical structures in MIDI data generated from ABC files. Further, their system uses

knowledge learned about musical structure to generate new melodies. Firstly, they take a corpus of 435 reels transposed into the same key and generate MIDI from these tunes. They then use a meter extraction algorithm which returns a series of timelags corresponding to multiple levels in the metrical hierarchy. The melodies and the meters are fed into a special type of recurrent neural network called a Long Short Term Memory Network (LSTM) which is trained to predict over all possible notes at time t using as input the note (and chord) values at time t −1. The authors claim the advantage of using a LSTM network is that it can learn long timescale (global) musical structure. They generate new tunes by presenting the network with the first few notes of a new tune (not from the corpus) and using it to predict the subsequent notes. They subjectively claim that their system generates "new and interesting" melodies, but present no experimental validation of the generated melodies.

Our system uses a simpler approach to generating new melodies that takes advantage of the fact that the input corpus is a text based markup language for musical scores. This facilitates the use of string comparisons in the analysis of the structure of the scores in our corpus. We also present an evaluation by 29 domain experts of the melodies generated using this approach.

4 A Pattern-based Sampling Approach

Figure 2 is a high level diagram describing the processes in our system.

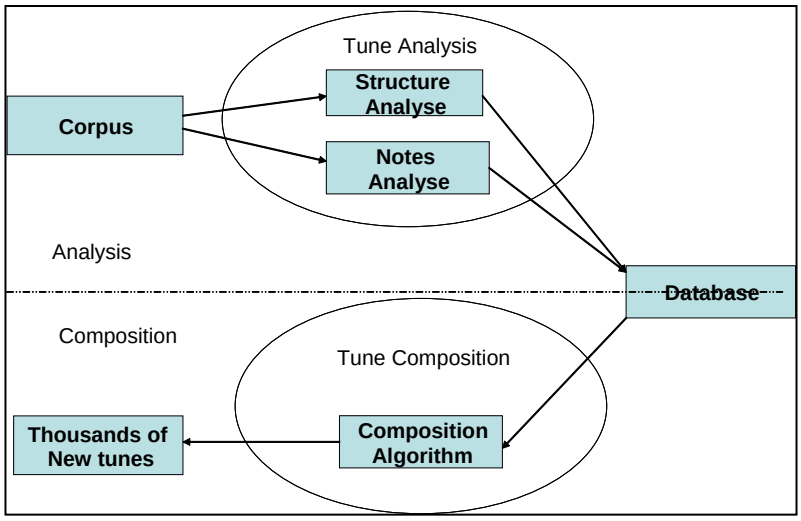


Figure 2: High level diagram of the composition system

In the analysis of tunes in the corpus, the system looks at both the structure of each tune and also the note sequences in each tune. We first apply an algorithm to reveal repeated patterns and these pattern structures will be used later during the stochastic sampling process used to generate new tunes. This is one way to overcome the limitations with purely n-gram models of music composition [12].

For each tune, we create an abstraction of the tune based on reoccurring tokens in the tune. There are various ways in the ABC language to represent repeats and these are considered by the system in generating a structure. Each half bar (4 quaver sequence) is considered to be a token.

```
  BG~G2 BGcG|BG~G2 Bdgd|BG~G2 BdcB|1 ADFG ABcA:|2 AGFG ABcA||
  ~g3d BGBd|~g2eg faaf|g2gd BddB|ADFG ABcA|
  ~g3d BGBd|~g2eg fa~a2|bgaf gedB|AGFG ABcA||
  Bdgd Bdgd|Bdgd BG~G2|Ac=fc Acfc|Ac=fc BG~G2|
Bdgd Bdgd|Bdef ~g3a|bgaf gedB|AGFG ABcA||
```

```
1 2 |1 3 |1 4 |5 6 |
1 2 |1 3 |1 4 |7 6 |
8 9 |10 11 |12 13 |5 6 |
8 9 |10 14 |15 16 |7 6 |
3 3 |3 1 |17 18 |17 1 |
```

Figure 3: Original ABC notation of the tune "The Flogging Reel" and the structure of the tune based on reoccurring 4-gram tokens

We assign a number to each unique token in the tune body. The first token number is 1. If the second token differs from the previous token then the number 2 is assigned to it. The algorithm then compares the third token with the previous two and if it is the same token as a previous token, it is assigned the same number. If not, a new number will be assigned and so the algorithm continues until the end of the tune is reached.

Symbol	Interpretation
BG~G2	A roll. ~ ignored.
Ac=fc	An accidental. = is ignored
(3BAG dB cAFA	A triplet, three notes in the time of two. (3BAG is considered to be a block that takes up two places. (3BA should not be separated
BG~G2	G2 is a G played with a length of two notes, so that it should not be separated. This means the G2 should be used as one block and takes up 2 places.
C'ABA	c' is considered to be one note and should not be separated from the note before it.
D,ACA	D, is considered to be one note and not separated from the note before it

Figure 4: ABC Language features that complicate the generation of n-grams

The whole tune is then converted to a sequence of token numbers. This abstracts the tune structure from the notes played in the tune. As some of the structures in the corpus are shared by several tunes we generate roughly 800 tune structures. Figure 3 is an example of the original ABC notation for the tune "The Flogging Reel" and the structure generated by our analysis. The algorithm then stores the tunebook location and name, the number of the tune in the tunebook, the name of the tune and the abstract structure represented by a sequence of token numbers in a database.

We then analyse note sequences in the corpus using n-grams. With different n values, n-gram analysis will generate note combinations for every possible combination of n notes in a tune. There are additional symbols in the ABC language that complicate the generation of n-grams. These symbols are summarised in Figure 4.

Along with the n-gram, the key of the source tune that the n-gram occurred in is also stored. This is because the key of the source tune will determine the actual note combinations in the tune. The key is then an identifier to identify musical phrases that occur in a particular key.

The system is flexible and supports any value of n but currently for ease of composition, the system counts 4-gram musical phrases. With n value equals to 4, there are 220,000 4-grams in the corpus. Figure 5 is an example of n-grams generated from the start of the tune "The Flogging Reel".

ID	GramContent	GramN	Key
181138	BGG2	4	Gmix
181139	GG2B	4	Gmix
181140	G2BG	4	Gmix
181141	BGcG	4	Gmix
181142	GcGB	4	Gmix
181143	cGBG	4	Gmix

Figure 5: Example n-grams generated from the start of the tune "The Flogging Reel"

To generate a new tune, the system first takes a structure at random from the database. Based on the key of the structure, the algorithm fetches an appropriate number of n-grams from the database and fills in an n-gram token into each token in the sequence. For each structure, a user configurable number of new tunes can be generated.

5 Evaluation

In order to evaluate our approach, we generated one hundred tunes and selected nine at random from the generated tunes for testing purposes. To the nine generated tunes, we added one human composed tune. MIDI renderings of the tunes were created so that subjects could listen to the tunes. We created an online survey using phpESP and asked subjects to rate each tune on a scale from one (poor) to five (excellent) in three categories:

1. Originality – To what extent the tunes differed from tunes already in the repertoire of the subject.
2. Aesthetic value – Did the subject like the tune.
3. Correctness – Did the tune sound right or did the tune sound odd.

We asked each subject three additional questions:

1. How long they had been playing music for?
2. Which of the ten tunes was their favourite?

3. Which one of the ten tunes was composed by a human?

We also left a freeform field on the survey and invited subjects to comment on the test. We posted an invitation to complete the survey on two popular web discussion forums used by traditional musicians [13, 14].

Within two days, the survey had been filled out by twenty nine subjects and our invitation had triggered a lively discussion as to which was the human composed tune on one of forums.

The average rating for each of the ten tunes in each of the three categories is presented in Figure 6.

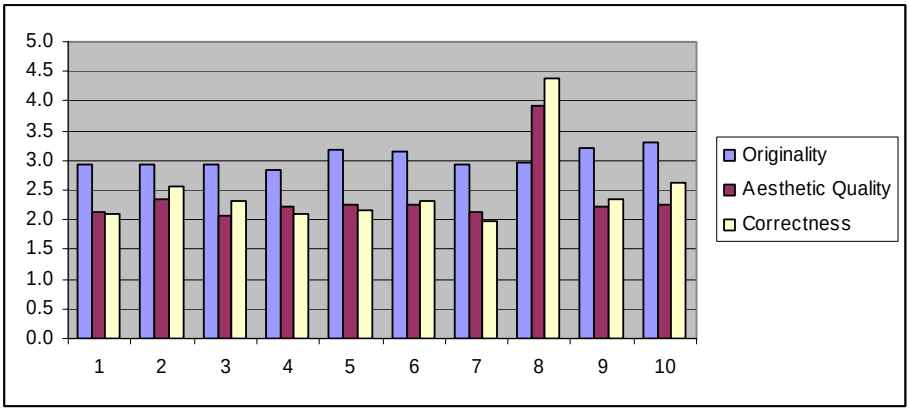


Figure 6: Average results for originality, aesthetic quality and correctness of the ten tunes in the test

On average, subjects rated the computer generated tunes 3 for originality, 2.2 for aesthetic quality and 2.3 for correctness. By contrast, subjects rated the human generated tune on average 3.0 for originality, 3.9 for aesthetic quality and 4.4 for correctness. This was not surprising as many of the computer generated tunes although they repeated in the correct places, contained unusually large pitch intervals that lent the tunes a somewhat atonal quality. This could have been corrected by "improving" the algorithm by filtering large pitch intervals, forcing the melodies to resolve or adding rules, however we judged that the tunes had a unique value due to this atonal quality that might be eliminated if the algorithm was too "constrained".

Most subjects (72%) correctly identified tune eight as being the human composed tune and similarly most subjects (58%) chose the human generated tune as their favourite from the selection. 13% of the subjects judged tune one to be their favourite, which was the favourite tune of the authors.

Most interesting from the survey was the feedback received in freeform comments submitted by subjects. Some of the subjects felt that many of the tunes combined interesting and aesthetically appealing phrases with atonal and unusual phrases. Pitch intervals in some of the tunes were unusually large for traditional tunes, with subjects commenting that many of the tunes had a modern feel reminiscent of modern Scots piping tunes. Many of the subjects commented that had the tunes been worked on by and played by a human rather than the MIDI rendering, the perceived imperfections could have been eliminated.

6 Conclusions

In this paper we presented our system that combined n-grams of notes together from a corpus into tune structures derived from the corpus to compose new tunes. We evaluated the generated tunes and had subjects try and tell the human composed tune from the generated tunes. From our evaluation it was clear that most subjects preferred the human composed tune to the tunes generated by our system. The tunes generated by our system, because they are structurally based on a tune from the corpus, usually repeat in the correct places and so often sound as though there was some planning in their construction. We feel that although many of the tunes have a strange atonal quality we are reluctant to dismiss them, because in our opinion some of the tunes possess interesting chord progressions and are curiously pleasing although they are in many ways different to what most listeners would classify as traditional reels. On the other hand we conclude that our use of random n-grams from the corpus needs further work if we are to improve the perceived correctness and thus the aesthetic quality of the tunes. The tunes used in this test can be listened to here: http://www.bryanduggan.com/phpESP/public/survey.php?name=GeneratedTunes.

References

1. Vallely F. The Companion to Irish Traditional Music: New York University Press; 1999.
2. Boden MA. Dimensions of creativity. Cambridge, Massachusetts: MIT Press; 1996.
3. Wallis G, Wilson S. The Rough Guide to Irish Music. First Edition ed. London: Rough Guides; 2001.
4. Vallely F. Flute Routes to 21st Century Ireland: National University of Ireland; 2004.
5. Walshaw C. The ABC home page. 2007 [cited; Available from: http://www.walshaw.plus.com/abc/
6. Mansfield S. How to Interpret ABC Notation. 2007 [cited; Available from: http://www.lesession.co.uk/abc/abc_notation.htm
7. Chambers J. O' Neills Books. 2007 [cited; Available from: http://trillian.mit.edu/~jc/music/book/oneills/
8. Norbeck H. ABC Tunes. 2007 [cited; Available from: http://www.norbeck.nu/abc/index.html,
9. Mozer MC. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing. Connection Science 1994.
10. Fernström M, Griffith N, Taylor S. BLIAIN LE BAISTEACH - SONIFYING A YEAR WITH RAIN. In: Proceedings of the 2001 International Conference on Auditory Display; 2001 July 29-August 1, 2001; Espoo, Finland; 2001.

11. Eck D, Lapamle J. Learning Musical Structure Directly from Sequences of Music. In: University of Montreal, Department of Computer Science, CP 6128, Succ. Centre-Ville, Montreal, Quebec H3C 3J7 Canada; 2006.
12. Conklin D. Music Generation from Statistical Models. Proceedings of the AISB 2007.
13. Chiff&Fipple. Chiff & Fipple Forums. 2007 [cited; Available from: http://chiffboard.mati.ca/
14. thesession.org. The session.org Forums. 2007 [cited; Available from: http://www.thesession.org

Using Computer Vision to Create a 3D Representation of a Snooker Table for Televised Competition Broadcasting

Hao Guo and Brian Mac Namee

School of Computing, Dublin Institute of Technology, Kevin St., Dublin 7, Ireland
{Brian.MacNamee@comp.dit.ie}

Abstract. The Snooker Extraction and 3D Builder (SE3DB) is designed to be used as a viewer aid in televised snooker broadcasting. Using a single camera positioned over a snooker table, the system creates a virtual 3D model of the table which can be used to allow audiences view the table from any angle. This would be particularly useful in allowing viewers to determine if particular shots are possible or not. This paper will describe the design, development and evaluation of this system. Particular focus in the paper will be given to the techniques used to recognise and locate the balls on the table.

1. Introduction

When watching televised snooker competitions it is often hard to see whether it is possible for a player to make a particular shot unless we can see the table from the player's point of view. Unfortunately, television cameras cannot always disturb the player to get the correct viewing angle. This paper will describe the Snooker Extraction and 3D Builder (SE3DB) a system developed to generate a 3D model of a snooker table from an overhead image of the table. This 3D model could be used in television broadcasting to allow viewers view the layout of the balls from any angle and so determine whether or not the tough shot the player is eyeing up is really possible.

Technology has been creeping into sports broadcasting more and more over the past number of years, and the following section will describe some notable examples of this to serve as a background for the discussion of our system. Next, the system itself will be described, including an overview of the digital image processing techniques utilized to extract the positions of the balls and a particular observation which makes this possible.

Finally, we will describe an evaluation that has been carried out on the system and suggest the directions in which we expect the work to go from here.

2. Background

Computer vision, coupled with technologies such as augmented reality (AR), has high potential for enabling a new class of application in television broadcasting [6]. Effects such as highlighting particular players, displaying team logos and illustrating distances have now become commonplace. One of the earliest examples of AR being used in sports broadcasting is the FoxTrax system [1]. This system was developed to highlight the location of the puck in televised ice hockey games. As the puck can be extremely hard for viewers to see when moving at high speeds across the ice, Foxtrax added a virtual glow and a comet trail to it. To achieve this, the developers of Foxtrax created an instrumented puck which emitted infra red pulses which were captured by cameras positioned around the ice rink. This system used 10 cameras, each of which focussed on a particular portion of the field, meaning that during a match, the puck was always in view of at least one camera. While the Foxtrax system was highly successful, drawbacks include the large amount of hardware required and the fact that modifications had to be made game equipment itself, which is not always possible.

The Hawk-eye system **(**www.hawkeyeinnovations.co.uk**)** [5] is another example of vision-based technology used in sports broadcasting. Originally developed for use in cricket matches, the Hawk-eye system creates a 3D virtual simulation of a sporting event which can then be played back and viewed from any angle to review the action. Since its original deployment Hawk-eye has gone on to be used in tennis and, more recently, snooker. Interestingly, in tennis Hawk-eye is not only used in broadcasting, but also by match officials to review calls on whether a ball is in or out.

Hawk-eye is particularly interesting in relation to the work being presented in this paper. Not only does Hawk-eye have similar goals to our own work (to create a virtual representation of a sports event) but it also focuses on the same sport (snooker). Unfortunately, as Hawk-eye is a commercial product there is very little information available in the literature about how it works.

As augmented reality has been used more and more in broadcasting, people have become more demanding in terms of quantity and quality of the visual information. Also, viewers want to interact with the content or influence the presented material on

TV. PISTE [2] is a system aimed at addressing all of these needs, providing broadcasters with the tools necessary to create enhanced content at transmission time, and the viewers with set-top-box technology capable of handing requests for interaction personalization. This system is capable of performing measurements, frozen shot display and comparison of different attempts displayed simultaneously.

3. Development

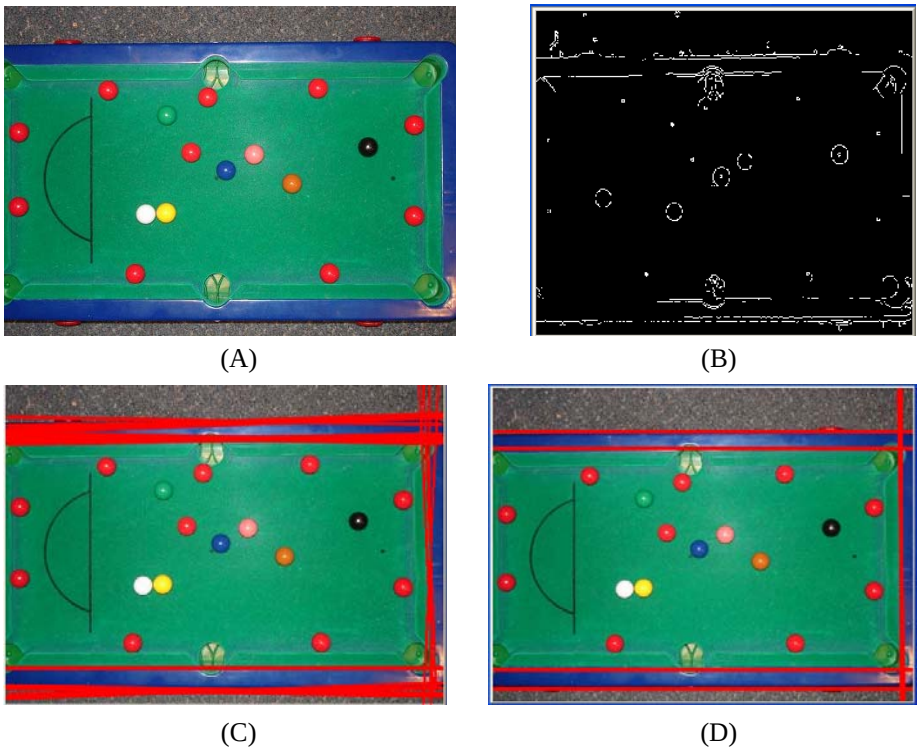
The purpose of this system is to extract information about the contents of a snooker table, and from this information build a virtual model of the table. Snooker is a particularly attractive sport for this kind of application as the contents of the environment are entirely controlled, and a large amount of information is fixed – e.g. the relative dimensions of the table and the balls. The development of our system involved the creation of a series of prototypes which attempted to create a virtual 3D model of a snooker table. This task can be divided into identifying the boundaries of the table itself and identifying the positions and colours of the balls on the table. The development of our system will be described in these two parts. Our system assumes that a human operator determines when a picture of the table should be taken and a subsequent 3D model created. The operator should only do this when the image of the table is clear of players, snooker cues, rests etc, and so we do not consider the presence of any of these objects.

In developing this system OpenCV (sourceforge.net/projects/opencvlibrary) was used for most of the computer vision algorithms used and OpenGL (www.opengl.org) was used for the display of the virtual table.

Extracting Table Boundaries

Detecting the boundaries of a snooker table is a relatively straightforward task as so much is known about what we expect to see in a typical snooker scene. In our system the boundaries of the table are extracted through edge detection using the Canny edge detector [7] followed by the Hough transform [3]. The results of a Hough transform is a set of candidate edges for the table boundary. From this candidate set of edges a representation of the table boundary can be extracted by using domain information about how we expect a table to appear. Figure 3 shows a series of images which

illustrate the steps required in identifying the table boundaries from an image of a snooker table.



(A) (B)

(C) (D)

Fig. 1. (A) The original image, (B) the edge detection result, (C) the results of a Hough transform super-imposed on the original image and (D) the final extracted table boundary super-imposed on the original image.

Identifying Snooker Balls

The first prototype developed to identify the balls on the snooker table used a very simple approach which performed an edge detection on the image of the table and used a circle-based Hough transform to extract the outlines of the balls. Unfortunately, the Hough transform did not perform particularly well. It is believed that the reasons for this are that the image was too noisy and that the circles extracted from the edges of the snooker balls were too far from perfect circles which caused problems for the OpenCV implementation of the Hough transform used.

The second prototype developed used a little more of the domain information available to us. This time it was realised that since the baize of the table was known to

be green it was obvious that the balls should stand out prominently from it. Hence, by using a simple flood fill algorithm [3, 4] the ball positions could be extracted without having to rely on edge detection. From this result, connected components can be identified as the balls on the table. Unfortunately, while some success was achieved using this technique, this prototype suffered from the fact that a number of the balls (in particular the blue and green balls) were particularly close to the colour of the baize and so were not identified successfully.

While developing the second prototype the key observation which resulted in the success of the final system was made. It was noticed that there is a strong specular reflection (a very bright spot) on the surface of each snooker ball. An example of this is shown in figure 2. This appears reliably in images of snooker tables, including those broadcast on TV. It was decided to take advantage of these bright spots to detect the balls using simple thresholding [3]. By converting the captured image of a snooker table into a grey-level image it is a straightforward task to set a threshold value which will extract only specular reflections (plus occasionally the white and yellow balls).

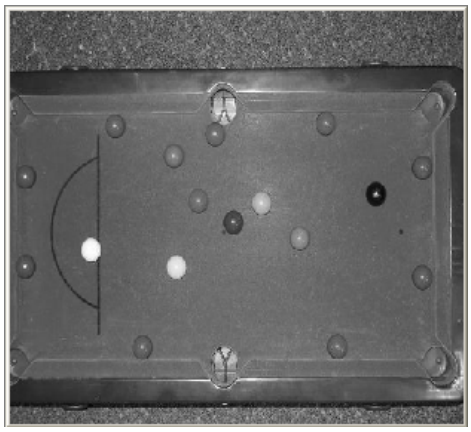
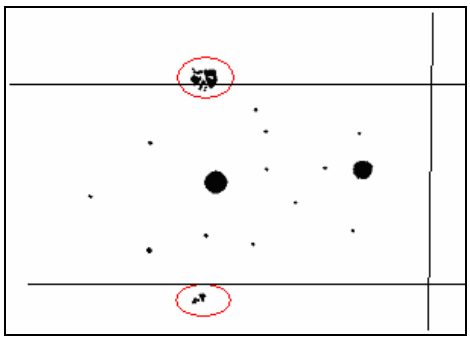


Fig. 2. A greyscale image of a snooker table which shows an obvious specular reflection on the surface of each ball.

Figure 3 shows an image of a snooker table, the results of thresholding and the final identified snooker balls. From figure 3 (B) it can be seen that, even though most of them are just tiny points, the specular reflections on the surface of each ball are particularly obvious. After the image is thresholded connected components analysis is used to extract each specular reflection, the centroid of which is considered to be the location of the ball. The white and yellow balls appear as particularly large objects in

the thresholded image. This is not a problem, however, as the centroid of these objects still gives a good position for the ball.

(A)



(B) (C)

Fig. 3. (A) The original image. (B) The results of thresholding. (C) The final set of detected balls highlighted with green dots.

From figure 3 (B) it can be seen that there are some spurious results in the thresholded image around the pockets of the table. These are the result of specular reflections arising from the material used to coat the insides of the pockets. These results were omitted from the final ball identification result (as shown in figure 3 (C)) by simply creating a rectangular region around each pocket in which specular reflections were ignored.

After detecting the positions of each ball, ball colours must be determined. Again the fact that the environment of a snooker table is closed is an advantage here as there are only a small set of possible colours that can be present (white, red, yellow, green, brown, blue, pink and black). The average RGB colour in the environs of each detected

ball is calculated and compared against a look-up table to determine the colour of each ball. The values in the look-up table were determined through experimentation.

Creating A Virtual Snooker Table

OpenGL (www.opengl.org) was used to create the virtual model of a snooker table used as part of this system. Figure 4 shows screenshots of an empty table and a table containing snooker balls which were created using simple OpenGL primitives. This is one aspect of the project which needs a considerable amount of further work as will be discussed in section 4.

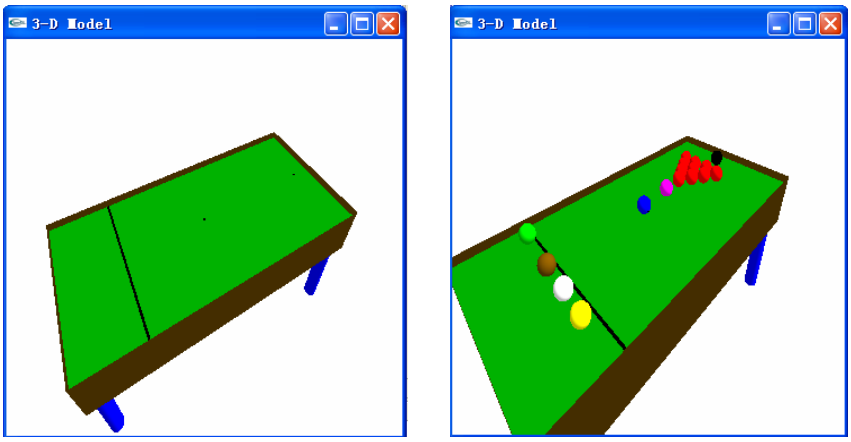


Fig. 4. The simple virtual snooker table used, shown empty and with balls.

4. Evaluation

In order to evaluate our system we performed a series of experiments in which various situations were set up on a small demonstration snooker table and the resultant virtual model produced by the system was compared against the real situation. This comparison sought to answer the following three questions:

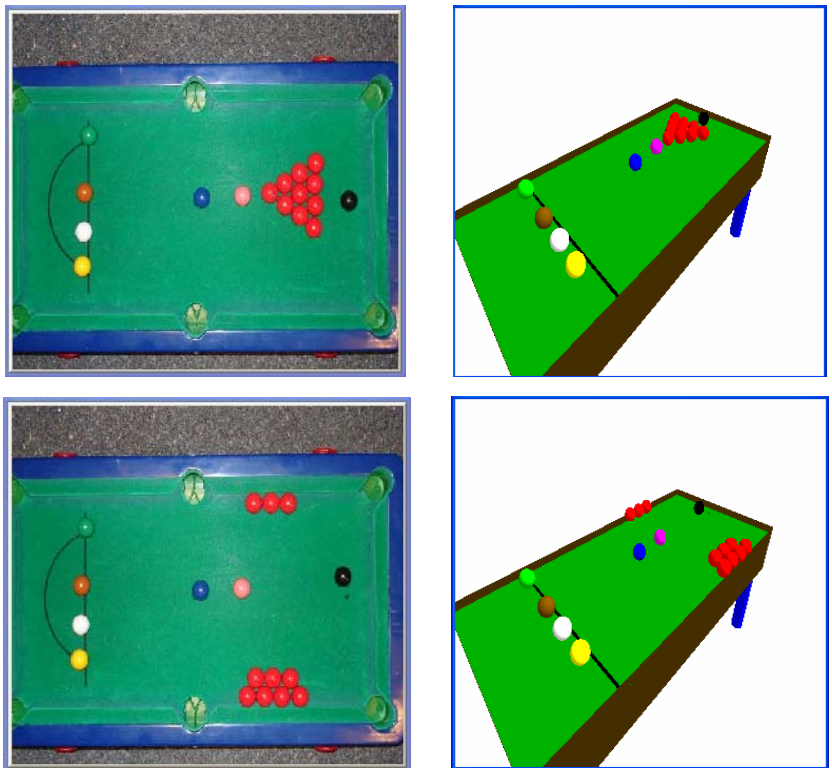
- Did the system recognise all of the balls?
- Did the system recognise any spurious non-existent balls?
- Did the system get all ball colours correct?

In this evaluation 13 situations were used. Figure 5 shows a selection of these evaluation situations and the resultant 3D models. As can be seen, both situations likely to arise in a typical frame of snooker, and more exotic situations were used.

In the 13 evaluation situations used a total of 181 balls appeared. In all of these situations only one ball was not detected. The reason for this was that the missing ball was placed too close to the pocket area. The system was designed to filter out balls located close to the pockets as these were expected to be spurious reflections caused by the material inside the pockets. The solution to this is to use a more realistic shape around the pocket region rather than the rectangle used at present.

Over the course of the 13 situations evaluated two spurious extra balls were detected. The cause of these was dirt lying on the surface of the table. It is believed that by adding some more domain information to the detection system these mistakes could be avoided – or possibly the table could be kept clean!

Colour detection performed reasonably well, only making a mistake in 3 out of 181 detection tasks. The performance of the colour detection system could very easily be improved by reviewing the look-up table and making it more comprehensive.



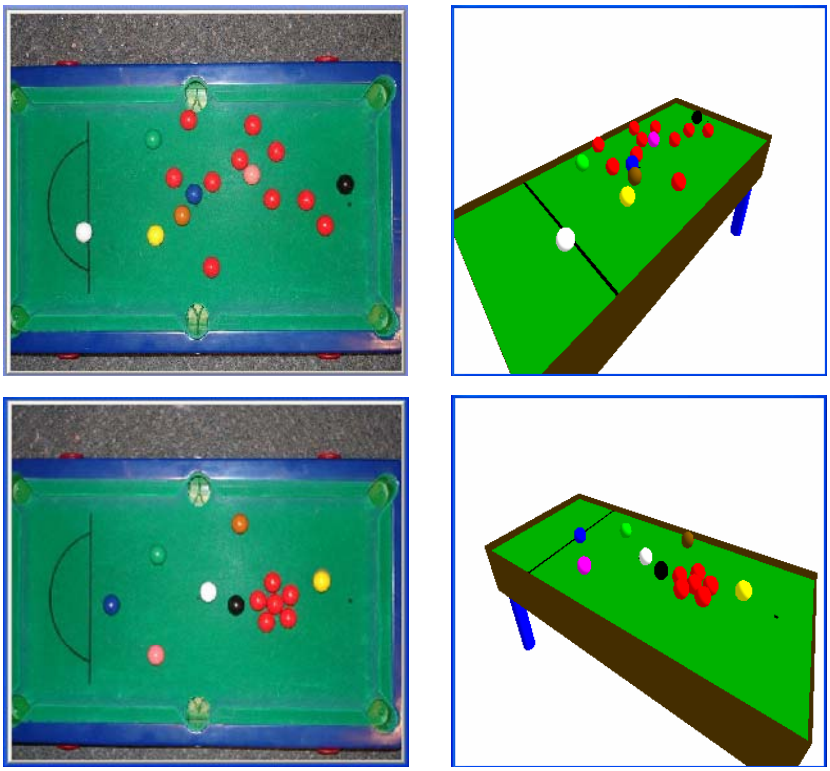


Fig. 5. Examples of images of a snooker table and the resulting virtual representation.

5. Conclusion

This paper has described the design, development and evaluation of the Snooker Extraction and 3D Builder (SE3DB) system, a system created to create a virtual 3D representation of a snooker table from a single overhead image of the table. The aim of the system is to render a 3-D model which can be used in televised snooker competition broadcasting to allow audiences view a rendering of the table from any angle. In developing the system, the key observation which made it possible was the realisation that each ball has a strong specular reflection that is very easily identified. The system has been evaluated and in most cases performs to an acceptable level of accuracy.

In the future we would hope to improve the system in the following ways:

- Create more realistic pocket regions so that balls placed next to pockets will not be filtered out as noise.

- Improve the colour detection system by improving the quality of the look-up table used to assign identified balls a colour.
- Create a much more visually appealing table model for the virtual 3D table renderings.
- Perform evaluations on a genuine competition snooker table.
- Add a calibration step to the system so that the system will cope with a camera positioned at any position and differences in specular reflection positions for balls on different parts of the table.

References

[1] Cavallaro, R. “The FoxTrax Hockey Puck Tracking System”, IEEE CG&A. vol. 17, no. 2, March/April, 1997

[2] Demiris, A. M.; Traka, M.; Reusens, E.; Walczak, K.; Garcia, C.; Klein, K.; Malerczyk, C.; Kerbiriou, P.; Bouville, C.; Boyle, E.; Ioannidis, N., “Enhanced sports broadcasting by means of augmented reality in MPEG-4”, In Proceedings of the International Conference on Augmented Virtual Environments and Three-Dimensional Imaging, 2001.

[3] Gonzalez, R.C.; Woods, R. E .“Digital image processing”, Prentice Hall, 2002.

[4] Khudeev, R., “A new flood-fill algorithm for closed contour”, In Proceedings of the IEEE International Siberian Conference on Control and Communications (SIBCON '05), pp172–176, 2005.

[5] Owens, N.; Harris C.; Stennett, C. “Hawk-eye tennis system”, In Proceedings of the International Conference on Visual Information Engineering: Ideas, Applications, Experience Visual Information Engineering (VIE2003), pp182-185, 2003.

[6] Pingali, G.; Opalach, A.; Jean, Y., “Ball Tracking and Virtual Replay for Innovative Tennis Broadcasts”, Pattern Recognition, In Proceedings. of the 15th International Conference on Pattern Recognition, Volume 4, pp152-156, 2000.

[7] Sharifi, M.; Fathy, M.; Mahmoudi, M.T.; “A classified and comparative study of edge detection algorithms”, In Proceedings of the International Conference on Information Technology: Coding and Computing, pp117–120, 2002.

A Kelly Criterion Approach to Dynamic Algorithm Portfolio Balancing

Alan Holland

Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork, Ireland.
a.holland@4c.ucc.ie,
http://www.4c.ucc.ie/~aholland

Abstract. In this work we posit that the Kelly Criterion for maximising compound growth in investments can be used to optimise dynamic rebalancing of algorithm portfolios. We use pari-mutuel gambling as a metaphor for determining how much CPU resources should be invested in various algorithms during execution. In this model we rely on information gleaned from past performance of these algorithms on similar problems and also current performance on the problem instance. In gambling parlance this lets us determine the *edge* and the *odds* for each algorithm during a given time period. We derive a formula for optimal investment of CPU resources so that the long-term expected search rate of a deterministic algorithm is maximised. Therefore, the main objective is to accelerate a systematic search technique as much as possible by allocating some of its CPU resources to non-deterministic search when the expected future gains outweigh the short-term costs in terms of time.

1 Introduction

Improving solution techniques for hard computational problems presents one of the most critical areas of research in computer science. Such problems pervade a diverse range of fields that include the frequency channel assignment [11], auction clearing [7, 18, 17], and determining the shortest route for a traveling salesperson [5, 1, 13]. Solution times for such $\mathcal{NP}$-complete problems may grow exponentially in the size of the problem. This militates against solving large instances in a timely manner. Many non-deterministic search algorithms involving intelligent heuristics have been developed to address this difficulty so that good quality solutions can usually be found quickly. However, guarantees of optimality may not be achieved in such a context. Furthermore, randomisation can lead to large variability in the performance of the search algorithm [4]. It often occurs that there is a low correlation between the performance of different search techniques. This offers the prospect of combining various search mechanisms so that we form a portfolio of algorithms amongst which computing resources are shared so that there is a balance between risk (the variability in search performance) and reward (the expected search performance). Our work investigates dynamic

algorithm portfolio balancing during run time so that we can allocate computing resources to those algorithms that will induce the best short-term gain so that the expected long-term performance over some fixed period of computation can be maximised. The rationale for this is that there are dependencies between the performance of algorithms. For example, if a local search algorithm finds a near optimal solution, this can dramatically improve the search rate of a deterministic search that makes use of the lower bound to backtrack from provably non-optimal sub-trees.

This paper is organised as follows. Section 2 describes previous work in algorithm portfolio design. Section 3 presents our approach and describes the Kelly criterion for weighting bets in pari-mutuel gambling events. This is used as a metaphor for our computing-resource allocation scheme that is described therein. Sections 4 and 5 discuss necessary future work to further evaluate our proposed framework and concluding remarks, respectively.

2 Related Work

Previously, Huberman *et al.* [9] developed a theory of algorithm portfolio design that employed an economics-based approach in an effort to balance risk and reward. Theirs was a general method for combining existing programs in a *static* portfolio so that the combinations were unequivocally superior to any of the individual algorithms. They employed Modern Portfolio Theory, as described by Harry Markowitz [15] to model the *efficient frontier*. An efficient portfolio is one that has the highest possible reward for a given level of risk, or the lowest risk for a given reward. Figure 1 illustrates an example of an efficient frontier with a bold red line.

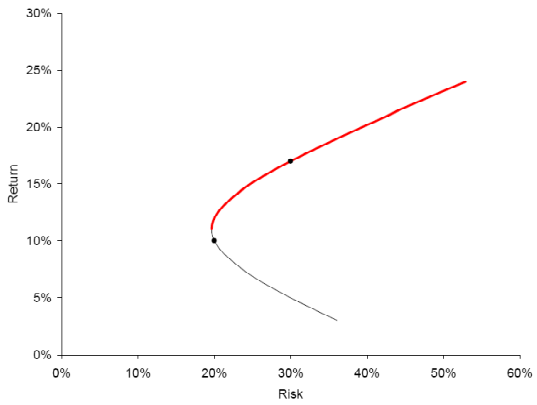


Fig. 1. An efficient frontier describing the weighting of assets in a portfolio of financial assets[19].

In all of these cases, the unpredictable variation in performance can be characterized by a distribution describing the probability of obtaining each possible performance value. The mean or expected values of these distributions are usually used as an overall measure of quality. They outline how expected performance is not the only relevant measure of the quality of an algorithm. The variance of a performance distribution also affects the quality of an algorithm because it determines how likely it is that a particular runs performance will deviate from the expected one. However, Markowitz's Model does not provide any indication of how to best exploit information gained during search. Dynamic re-balancing can be more effective because additional information can be used to good effect.

Previous work on dynamic algorithm selection includes Horvitz *et al.*'s algorithm recommendation based on the performance of the candidate algorithms during a predefined amount of time, called the observational horizon [8]. In anytime algorithm monitoring [6], the dynamic performance profile of a planning technique is updated according to its performance, in order to stop the planning phase when further improvements in the actions planned are not worth the time spent in evaluating them. Using a reinforcement learning approach, Lagoudakis and Littman [12] showed how algorithm selection can be formulated as a Markov Decision Process with sequences of recursive algorithms formed dynamically at run-time. A variation of Q-learning is used to find an online algorithm selection policy. Most prior work on algorithm portfolios focus on choice criteria for building the set of candidate solvers whose performance on specific instances exhibit low correlation. The portfolio is then executed in parallel or used as a pool of potential algorithms for selection [14].

3 Our Approach: Betting on Algorithm Performance

We argue that just as fund managers continually balance risk and expected reward in a portfolio of financial assets, so too should an algorithm portfolio respond to events and re-balance accordingly. The requirement for re-balancing in algorithm portfolios is even greater because of the dependencies between the outcomes of some search techniques. Pertinent events during a search process include the discovery of tighter upper or lower bounds on the optimal solution. The effectiveness of some algorithms is greatly enhanced by such discoveries whilst the performance of some search heuristics may be unaffected. Therefore, following the discovery of useful information it makes sense to increase the computational resources assigned to algorithms that benefit from such information. The key question remains as to how we should divide our computing resources amongst the possible algorithms.

We draw an analogy between pari-mutuel gambling on a random event such as a horse race and weighting ones' computational resources on the probabilistic performance of algorithms. The goal in pari-mutuel gambling is to judge the true probabilities of different outcomes more accurately than the implied probabilities of the odds provided by the bet-takers.

3.1 Pari-mutuel Betting System

Pari-mutuel betting is most commonly found in horse racing so we use this setting in Example 1 to describe how it works.

Example 1 *Consider a horse-racing event with four runners and thus four possible winners as outcomes. Each horse has received a certain amount of backing, or money wagered:*

- *Horse 1 $130.00*
- *Horse 2 $270.00*
- *Horse 3 $50.00*
- *Horse 4 $450.00*

Thus the total pool of money wagered on this horse race is $900.00. Once the race has started, bets are stopped. Once the event is finished and, say, the winning outcome is determined to be Horse 3 with $50.00 wagered. The payout is calculated as follows. First the commission for the "bookie" or company accepting bets is deducted from the pool. For example, given a commission rate of 15% the remaining pool of money to be distributed among winning bettors is: $\$900 - (0.15 * \$900) = \$765$. *In this case, this money is now distributed amongst those who bet on Horse 3:* $\$765/\$50 = \$15.30$ *per $1.00 wagered. So Horse 3 is said to pay out $15.30.*

We draw an analogy between the likelihood of a horse winning a race and the likelihood of an algorithm, a_i, discovering valuable information in a time window t_j. We assume that these time windows are short so that the probability of such is a discovery is directly proportional to the time invested in the algorithm. We also assume that the search rate of deterministic algorithms depends on present information, *i.e.* a tighter bound on the optimal solution will increase the search rate.

3.2 The Kelly Criterion

The Kelly Criterion is a technique best known in gambling and finance for maximising the long-term growth rate of repeated plays of a given gamble that has positive expected value. John L. Kelly first described the technique in a 1956 issue of the Bell System Technical Journal [10]. More precisely, the formula specifies the fraction of an existing monetary fund (or bankroll) that should be invested (or wagered) during each instance of a game. In addition to maximizing the long-term growth rate, the formula allows for a zero risk of ruin in theory. This is only true, however, when there is no minimum bet amount and the currency is infinitely divisible.

The Kelly Criterion is sometimes referred to as *Fortune's Formula* and has been used to maximize returns from stock market investments [16]. If you are betting on a football match or horse race in which you have some inside information you would seek to maximize this advantage that involves more accurate

knowledge of the true probability and odds such that each bet has a positive expectation. The Kelly Criterion is a formula for determining the optimal percentage of the bankroll that should be invested in this gamble, given the objective is to maximise compound growth over many such events without exhausting the bankroll.

For simple bets with two possible outcomes, one incurring a complete loss of the entire wager and the other involving winning the bet amount multiplied by the payoff *odds*, the Kelly Criterion specifies that the optimal fraction, f_K^* of the current bankroll to wager is:

$$f_K^* = \frac{p(v+1)-1}{v}, \tag{1}$$

where v is the odds received on the wager, p is the probability of winning and q is the probability of losing (1 - p) in this case.

Example 2 *Consider a gambler with a bankroll of \$100. He receives odds of* 5-*to*-1 *on a horse winning a race. Given that the gambler has received information that is not publicly available, he estimates the likelihood of the horse's success at* 35%. *The optimum fraction of his bankroll he should place on this gamble is*

$$f^* = \frac{(0.35 * 5) - 0.65}{5} = 0.22.$$

So, given that his bankroll is \$100, the gambler should invest \$22 in this bet to maximise expected compound growth in his wealth according to the Kelly Criterion.

Note that the Kelly criterion allows one to bet on multiple different winners if more than one horse has a positive edge over the listed odds.

3.3 Dynamic Algorithm Portfolio Design using the Kelly Criterion

We use the pari-mutuel gambling metaphor to decide how we distribute our resources (computing time) amongst various non-deterministic algorithms whose performance in the event (search instance) is probabilistic. Any CPU resources not invested will instead be used by a deterministic search algorithm that seeks an optimal solution. Firstly, we outline what aspects of algorithm performance that are relevant for this metaphorical viewpoint. Axiom 1 makes the assumption that if new information is supplied to a deterministic search algorithm, backtracking is possible at an earlier stage in the search tree. The axiom supposes that the search rate in terms of nodes per second improves in proportion to the amount of information supplied.

Axiom 1 *New information, I, provided by the results of a non-deterministic search algorithm (*e.g. *a new constraint) can increase the search rate of a deterministic search algorithm. We assume the increase in search rate, ΔS, is directly proportional to the amount of information provided,*

$$\Delta S \propto I.$$

Axiom 2 assumes that we can distribute computing resources in as granular a manner as we wish. Modern CPU management facilities allow program developers to allocate resources for processes in an arbitrary manner [2].

Axiom 2 *Computing resources are infinitely divisible and can be shared among an arbitrary number of search processes.*

The currency in use is search rate (nodes per second) by time (seconds) whose units are therefore nodes in the deterministic search tree. When we divert our search effort to a non-deterministic algorithm we are doing so at the expense of searching more nodes in a tree systematically. By diverting some CPU resources to a non-deterministic search we slow the systematic search in the hope that valuable information is attained that will accelerate the deterministic algorithm in subsequent time periods. However, as the deterministic search rate increases the investment in time becomes more expensive. For now we wish to know how much time we should invest in a set of search techniques so that the long-term compound search rate of a deterministic algorithm is maximised.

The Kelly Criterion specifies the fraction of an existing monetary fund that should be invested or wagered during each instance of a game. For the case of dynamic algorithm portfolios we examine how CPU resources should be distributed amongst a portfolio of algorithms at regular time intervals. In this portfolio we assume that there is a set of non-deterministic algorithms and a single deterministic algorithm whose performance can be improved via the outputs from the other algorithms. The problem is to decide how much time to invest in utilising these algorithms that alone will never provide a guarantee of optimality but can enhance the performance of systematic search.

Let us first assume that the deterministic search rate is S nodes per second. We assume that from prior analysis of similar problems that we can learn relevant values that guide our investment decisions. In particular, during any time period t_i the execution of the various algorithms we need to inform our portfolio rebalancing routine about the probabilities p_{ik} of each search algorithm finding useful information for the deterministic search. In other words, for each time period t_i we require the probability of a "win" for each algorithm k. A win corresponds to finding new information that accelerates deterministic search, such as an improved bound or a learned constraint.

In this work we assume this is a blackbox entity comprising a previously learned prediction system such as a neural network or bayes net. Further exposition of this entity is left for future work. This system will also need to inform us of the likely significance of any additional information attained from a search routine in that time period. The odds generated are a function of the value of the information gained with respect to the value that could be gained from deterministic search. As it is repeatedly used in a dynamic manner during search it should make use of instance specific performance up to that point. This facilitates diversification of search so that the initial focus is on those non-deterministic search techniques with the best expected performance and then gradually switches to those that return useful information less often but whose performance exhibits low correlation with previously used techniques. Thus, we can take advantage of

the heavy-tailed distribution behaviour of search performance as described by Gomes *et al.* [3]. The advantage of our dynamic approach is that it can be done in a reactive manner when necessary rather than committing to a computing schedule in advance of run time.

We denote the expected improvement in the systematic search rate due to information provided by the non-deterministic search routine k in time period t_i as v_{ik}. In betting terms, this is the odds as expressed $v - to - 1$. In algorithm portfolio terms this is the expected gain relative to that of the deterministic search. If the probability of algorithm k providing information is low and the expected value of this information is not attractive, then it is likely that we will not wish to invest any CPU resources in this routine. However, as time passes and the results of other more promising non-deterministic algorithms prove to be unsuccessful our blackbox advisory system may inform us that due to the low correlation in performance it may become an increasingly attractive possibility.

Theorem 1. *Given Axioms[1-2], the fraction of CPU resources during time period t_i diverted to non-deterministic search algorithm k in order to provide information that maximally increases the compound growth in the deterministic search rate is given by*

$$f_{ik}^{*} = \frac{p_{ik}v_{ik} - 1}{v_{ik} - 1}.$$

Proof: We take a bet (or investment of search effort), $f \times S$ for time period t_i, with odds of v-to-1. This means that if the algorithm finds useful information in that period, the fractional improvement in the deterministic search rate is linear in the amount of time invested in that search should there be an improvement. The bettor's search rate is adjusted as follows.

$$S' = \begin{cases} S \times (1 + (v-1)f) & \text{in the event of a successful search} \\ S \times (1 - f) & \text{in the event of an unsuccessful search,} \end{cases}$$

This differs slightly from the betting metaphor because the original wager is not returned. The CPU resources expended on finding information relevant to aid the deterministic search cannot be recovered. If the above process is repeated n times we say there are w occasions when the bettor wins and, therefore, $n - w$ instances where losses occur. The new search rate is

$$S' = S[(1 + (v-1)f)^{w} * (1 - f)^{n-w}].$$

Hence, the increase in search rate per individual betting event is

$$g = \sqrt[n]{\frac{S'}{S}} = \sqrt[n]{(1 + (v-1)f)^{w}(1 - f)^{n-w}}.$$

To maximise the expected growth in the search rate we choose a value of f so that $g(f)$ is maximised. This is also the same value for which $\ln|g(f)|$ is maximised.

$$\ln|g(f)| = \frac{1}{n}[w \ln|1 + (v-1)f| + (n-w)\ln|1 - f|]$$

We differentiate this equation and set to zero:

$$\frac{\delta \ln |g(f)|}{\delta f} = \frac{(v-1)w}{n(1+(v-1))f} - \frac{n-w}{n-nf} = 0.$$

Solving for f^* to maximise the growth in search rate, we find

$$(v-1)wn - nf^*(v-1)w = n^2 - nw + n^2(v-1)f^* - nw(v-1)f^*,$$

$$f^* = \frac{wv-n}{n(v-1)}.$$

Letting $\frac{w}{n} = p$, the probability of a win event,

$$f^* = \frac{pv-1}{v-1}. \tag{2}$$

Equation 2 specifies the proportion of CPU resources during a particular time period to dedicate to a particular non-deterministic search algorithm. So, in general, for time period t_i given inputs v_{ik} and p_{ik} from a blackbox system the regarding expected performance for algorithm k the optimal Kelly CPU allocation is

$$f^*_{ik} = \frac{p_{ik}v_{ik}-1}{v_{ik}-1}, \tag{3}$$

■

We see from Equation 3 that it is very similar to that of the Kelly Criterion (Equation 1). The key difference is that the odds returned are 1 less than that in Kelly betting. This occurs because the original wager (investment of CPU resources) is not returned following a successful search by a non-deterministic algorithm.

4 Discussion and Future Work

The Kelly system in practice has advantages and disadvantages. In betting and financial circles its usage guarantees that you will never lose all your bankroll (by assuming an infinitely divisible currency and no minimum bet amounts). However, it does not guarantee that you will not lose money. The chance of the bankroll of dropping to $\frac{1}{n}$ of the original is $\frac{1}{n}$. In terms of algorithm portfolios, this means that there is considerable volatility in the attained search rates even though on average they are maximimsed. Possible ways of reducing such volatility is to bet "fractionally-Kelly". The expected rate of compound growth is reduced but volatility is also significantly reduced.

There are many possible directions for future work in terms of this nascent research topic. Firstly, there is the key issue of designing the blackbox advisor system to inform the Kelly portfolio rebalancing of algorithms' probabilities of discovering useful information and its likely value. When setting the odds

on returns, these must be benchmarked according to the expected returns for the deterministic algorithm. In case the expected value of the information gain for a non-deterministic algorithm is inferior, then Kelly betting stipulates that no investment is made in that search routine. We plan to investigate different approaches to designing such a system that will possibly incorporate machine learning techniques such as neural networks or bayes nets to determine values for p_{ik} and v_{ik}.

We also intend to conduct an experimental study of how such a system performs in practice and compare performance with individual algorithms and static portfolios of algorithms. This is an important step in validating the aforementioned theory.

5 Conclusion

Dynamic algorithm portfolio balancing is an important area of research in computer science as it addresses one of the foremost research problems, that of minimising the time required to solve $\mathcal{NP}$-complete problems optimally. Our objective is to maximise the compound growth in search rate for a deterministic algorithm with the aid of non-deterministic search routines that provide information that can accelerate this search. However, the computing resources must be shared, thus leading to a competition.

We derived a formula for deciding on the maximally aggressive strategy for increasing the rate of search for a deterministic algorithm via information gain from non-deterministic algorithms. This approach is similar to that of maximally aggressive betting that adopts to Kelly criterion. Our approach requires inputs from historical data regarding the probabilistic performance of such search routines and the added value of the information at different time periods. Although the approach is as yet untested and further work is required to empirically verify its applicability, we believe that its novelty constitutes a significant contribution to the important field of algorithm portfolio design and opens up many interesting avenues for further study.

References

1. N.L. Biggs, E.K. LLoyd, and R.J. Wilson. *Graph Theory 1736-1936*. Clarendon Press, 1976.
2. Walter Binder and Jarle Hulaas. A Portable CPU-Management Framework for Java. *IEEE Internet Computing*, 08(5):74–83, 2004.
3. Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems. *Journal of Automated Reasoning*, 24(1):67–100, 2000.
4. Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting Combinatorial Search through Randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, pages 431–437, Madison, Wisconsin, 1998.
5. G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer, 2006.

6. Eric A. Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: a dynamic programming approach. *Artif. Intell.*, 126(1-2):139–157, 2001.
7. Alan Holland. *Risk Management for Combinatorial Auctions*. PhD thesis, University College Cork, Ireland, July 2005.
8. Eric Horvitz, Yongshao Ruan, Carla P. Gomes, Henry A. Kautz, Bart Selman, and David Maxwell Chickering. A bayesian approach to tackling hard computational problems. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 235–244, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
9. Bernardo E. Huberman, Rajan M. Lukose, and Tadd Hogg. An Economics Approach to Hard Computational Problems. *Science*, 275:51–54, 1997.
10. John L. Kelly Jr. A new interpretation of information rate. *Bell System Technical Journal*, 35:917–926, 1956.
11. I. Katzela and M. Naghshineh. Channel Assignment Schemes for Cellular Mobile Telecommunications: A comprehensive survey. *IEEE Personal Communications*, pages 10–31, 1996.
12. Michail G. Lagoudakis and Michael L. Littman. Algorithm selection using reinforcement learning. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 511–518, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
13. E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Khan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.
14. Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 556–572, London, UK, 2002. Springer-Verlag.
15. Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.
16. William Poundstone. *Fortune's Formula: The Untold Story of the Scientific Betting System That Beat the Casinos and Wall Street*. Hill and Wang, New York, 2005.
17. Michael H. Rothkopf, Aleksander Pekeč, and Ronald M. Harstad. Computationally Manageable Combinatorial Auctions. *Management Science*, 44(8):1131–1147, 1998.
18. Tuomas Sandholm. Algorithm for Optimal Winner Determination in Combinatorial Auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.
19. Paul Wilmott. *Paul Wilmott Introduces Quantitative Finance*. John Wiley & Sons, West Sussex, England, 2001.

The Evolution of a Kernel-Based Distance Metric for k-NN Regression

Tom Howley and Michael G. Madden

National University of Ireland, Galway,
thowley@vega.it.nuigalway.ie, michael.madden@nuigalway.ie

Abstract. k-Nearest Neighbours (k-NN) is a well understood and widely-used approach to classification and regression problems. In many cases, such applications of k-NN employ the standard Euclidean distance metric for the determination of the set of nearest neighbours to a particular test data sample. This paper investigates the use of a data-driven evolutionary approach, named KTree, for the automatic construction of a *kernel*-based distance metric as an alternative to Euclidean distance. The key idea behind this approach is that a different distance metric is generated for a particular data domain. The performance of k-NN with the standard Euclidean distance measure is compared with that of k-NN based on a kernel-based distance metric evolved by KTree. This comparison is based on experiments on both synthetic and real-world datasets.

1 Introduction

k-Nearest Neighbours (k-NN) is a well understood technique that is widely used in many classification and regression problems [1]. In many applications of k-NN, the Euclidean distance is used to determine the k nearest neighbours to a particular test sample, the resulting prediction depending directly on the particular set of neighbours chosen. As noted by Yu *et al.* [2], the conventional k-NN can perform well with non-linear problems, but loses its power with some complicated problems, especially when the sample distribution is arbitrary. However, if an appropriate kernel is chosen to reshape the distribution of samples, a *kernelised* k-NN algorithm may improve its performance. This is an example a kernel-based learning method, in which a kernel function is used to transform the original data into a new feature space. The choice of kernel and associated kernel parameters is a key step in the application of any kernel method, such as kernelised k-NN, to a problem. Previous research carried out by the authors demonstrated that an evolutionary approach, named KTree, was effective in the automatic construction of kernels in Support Vector Machine (SVM) classification [3]. This paper investigates the use of the KTree approach to evolve a kernel-based distance metric for k-NN regression.

The paper begins in Section 2 with an overview of kernel methods, kernel functions and the kernelised k-NN algorithm. Section 3 then describes the KTree algorithm. Experimental results and analyses are presented in Section 4. Section 5 evaluates research related to this work and Section 6 presents the main conclusions.

2 Kernel Methods, Kernel Functions & k-NN

In kernel methods for classification or regression, the kernel function is used to recode the data into a new feature space that may reveal regularities in the data that were not detectable in the original representation. This allows the use of algorithms based on linear functions in the new feature space; such linear methods are both well understood and computationally efficient. With kernel functions, no explicit mapping of the data to the new feature space is carried out – this is known as the *kernel trick*. A kernel function, $K(x, z)$, calculates the dot-product of two data samples, x and z, in the feature space, ϕ, that the kernel defines: $K(x, z) = \langle \phi(x), \phi(z) \rangle$

2.1 Kernelised k-NN

A machine learning algorithm may be "kernelised" by first reformulating the algorithm so that all data enters it in the form of dot-products of sample pairs. Each dot-product calculation in the algorithm is then replaced by a kernel function, thus transforming the algorithm into the feature space defined by that kernel. The classic example of a kernel method is the SVM [4]. However, other machine learning algorithms can be reformulated as a kernel method, one example being k-NN, a method that can be used in both classification and regression settings. There have been many variants of the k-NN algorithm, but the basic idea is as follows: the distance between the test sample and each sample in the training set is calculated to determine the k samples that are closest to the test sample; in classification, the majority class of these nearest samples (or nearest single sample when k = 1) is returned as the prediction for that test sample; in regression, the (possibly weighted[1]) average value of the dependent variable for the k nearest samples is returned as the prediction.

The k nearest samples are often determined using the Euclidean distance, $d = ||x - z||$. With kernel methods, the kernel's feature space is known as a *dot product space* and therefore has a naturally defined norm: $||x|| = \langle x, x \rangle$. Any norm defines a metric d via [5]:

$$\begin{aligned} d(x, z) &= ||x - z|| \\ &= \sqrt{\langle x - z, x - z \rangle} \\ &= \sqrt{\langle x, x \rangle + \langle z, z \rangle - 2\langle x, z \rangle} \end{aligned} \tag{1}$$

A kernel distance metric is therefore defined as:

$$d_K(x, z) = \sqrt{K(x, x) + K(z, z) - 2K(x, z)} \tag{2}$$

The above equation can be used to derive distance measures from any kernel, which can substitute the Euclidean distance measure in a k-NN algorithm[2].

[1] The k-NN implementation used for the experiments reported in this paper does not employ a distance-weighting mechanism.

[2] In this work, a distinction is made between the Euclidean distance measure and kernel-based distance measures. We use the term 'Euclidean distance' to refer to the conventional k-NN distance measure defined in the original input space and we use the term 'kernel-based distance' to refer to the Euclidean distance as defined in the kernel transformed space.

2.2 Kernel Function

As with any kernel method, a key step in the application of kernel k-NN is kernel selection. With SVMs, for example, typical choices for kernels are the Linear, Polynomial, RBF and Sigmoid kernels. One alternative to using these standard kernels is to employ a kernel that has been customised for a particular application domain, e.g. the string kernel of Lodhi *et al.* [6] and kernels for protein classification [7]. Whether building complex kernels from simpler kernels, or designing custom kernels, there are conditions that the kernel must satisfy before it can be said to correspond to some feature space. Firstly, the kernel must be symmetric, i.e. $K(x,z) = \langle \phi(x), \phi(z) \rangle = \langle \phi(z), \phi(x) \rangle = K(z,x)$. Typically, kernels are also required to satisfy Mercer's theorem, which states that the matrix $K = (K(x_i, x_j))_{i,j=1}^{n}$ must be positive semi-definite, i.e. it has no negative eigenvalues [4].

3 KTree

As previously highlighted, a critical stage in the use of kernel-based algorithms is kernel selection, as this can be shown to correspond to the encoding of prior knowledge about the data [8].

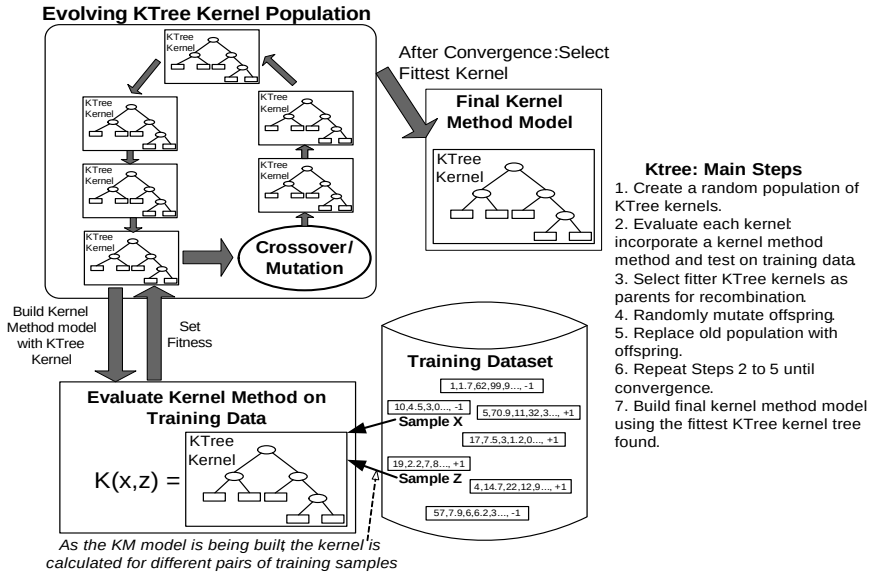


Fig. 1. KTree algorithm

Kernel method users can select from one of the standard kernels, construct new kernels using simpler kernels as building blocks, e.g. the kernel, $K(x,z) = K_1(x,z) + K_2(x,z)$, or customise a kernel for a particular problem. Ideally, a kernel is selected or customised based on prior knowledge of the problem domain, but it is not always

possible to make the right of choice of kernel *a priori*. KTree addresses this problem by using the evolutionary technique of Genetic Programming (GP) to discover a suitable kernel for a particular problem. KTree has been previously demonstrated with SVM classifiers [3], but this approach can be used with other kernelised pattern analysis algorithms. The aim of KTree is the discovery of new kernels that best represent the underlying data from a particular problem domain; in the context of kernel k-NN, KTree allows for the discovery of a new distance metric for a particular data domain. With KTree, a tree structure, known as a *KTree kernel* (see Figure 2) is used to represent a kernel function. The objective of KTree is to find a KTree kernel that best represents the data. An overview of the KTree algorithm is shown in in Figure 1.

3.1 KTree Kernel Representation

The KTree kernel used to represent a kernel function must take two data samples as inputs and provide a scalar value as output. An example of a KTree kernel is shown in Figure 2.

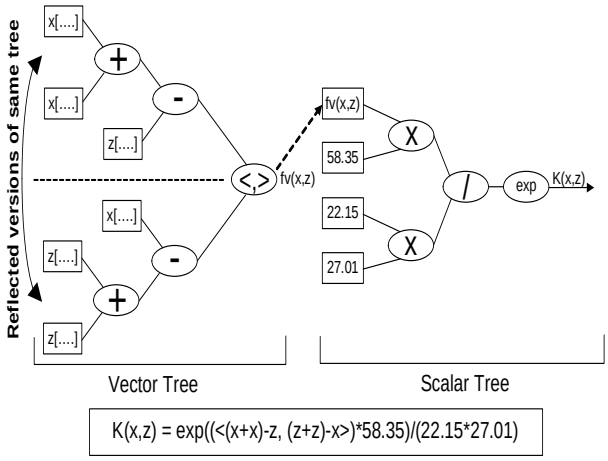


Fig. 2. Example KTree Kernel

The diagram shows that the KTree kernel is split into two parts, the vector and the scalar tree. The inputs to the vector tree are the two samples, x and z, for which the kernel is being evaluated. These inputs are passed through vector operators, such as *add* or *subtract*, which in turn pass vectors onto the next node. To ensure that the output of this tree is symmetric, the entire vector tree is evaluated twice, swapping the inputs x and z for the second evaluation. The final output of the vector tree, $f_v(x, z)$, is the dot product of these two evaluations. This output becomes an input, along with randomly generated constant terminals, for the scalar tree. This design was chosen to allow for the use of complex mathematical operators, such as exp and $tanh$, in the scalar tree. Applying these operators directly to the vector inputs could result in overly complex and unusable kernels. A second motivation for this design is that it is also capable of representing the standard kernels used in SVMs, e.g. the RBF kernel and Polynomial

kernel. Although symmetry is satisfied, this kernel design is not guaranteed to produce Mercer kernels. However, non-Mercer kernels are filtered out (see Section 3.2).

A specification of the KTree kernel is given in Table 1, showing the input terminals and operators used for the vector and scalar tree parts of a KTree kernel. The vector tree is a GP tree, where the input terminal set comprises the two vector inputs to the kernel function, x and z. The operators of the vector tree take two vectors as an input and return a single vector as output. A vector operator is calculated as follows:

$$[x_1, x_2, \ldots, x_m]\ op\ [z_1, z_2, \ldots, z_m] = [x_1\ op\ z_1, x_2\ op\ z_2, \ldots, x_m\ op\ z_m] \quad (3)$$

where op is one of the operators listed for the vector tree in Table 1 and m is the length of the vector inputs. For example, an addition in the vector tree is calculated as follows:

$$[x_1, x_2, \ldots, x_m] + [z_1, z_2, \ldots, z_m] = [x_1 + z_1, x_2 + z_2, \ldots, x_m + z_m] \quad (4)$$

The scalar tree of a KTree kernel is a GP tree, where the input terminal set comprises the output of the vector tree, denoted as $f_v(x, z)$, and a set of randomly generated constants. Note that $f_v(x, z)$ may occur multiple times in a scalar tree. The set of scalar operators (unary and binary) used in the scalar tree is listed in Table 1. Note that while the use of constants in the scalar tree of the kernel influences the decision boundary of an SVM classifier, it has no bearing on the ordering of neighbours for k-NN without distance-weighting.

Table 1. KTree kernel specification

Vector Tree	
Input Terminals:	$x[..]$, $z[..]$
Operators:	add, $subtract$, $multiply$
Scalar Tree	
Input Terminals:	$const$, $f_v(x, z)$
Operators:	add, $subtract$, $multiply$, $divide$, exp, $power$, $tanh$

As shown in Figure 1, the first step of the KTree algorithm is to create a random population of kernels. For this initial population, each KTree kernel (both vector and scalar parts) is generated by randomly creating a root node and by growing a tree from this node until either no more leaves can be expanded (i.e. all leaves are terminals) or until a preset *initial* maximum depth has been reached (2 for the experiments reported here). The evolutionary process shown in Figure 1 involves the application of mutation and crossover operators on selected KTree kernels. For mutation, a point in either the vector or scalar tree is randomly chosen and the sub-tree at that point is replaced with a newly generated tree (vector or scalar, depending on where mutation occurred). Mutation of individual nodes (e.g. constant terminals) is not employed. Crossover between two KTree kernels begins with the selection of a random point from either the vector or scalar part of the first KTree kernel. The location of the crossover point on the second

KTree kernel is constrained so that crossover does not occur between the scalar part of one KTree kernel and the vector part of another. Rank-based selection was employed for the selection of the candidates for crossover. To prevent the proliferation of massive tree structures, pruning is carried out on KTree kernels after mutation, maintaining a maximum depth of 12 (for either the vector or scalar part). A population of 500 KTree kernels was used for all experiments.

3.2 Fitness Function

As with any evolutionary algorithm, a key element of KTree is the choice of fitness function. In previous work on the use of KTree for SVM classification [3], the authors investigated a number of fitness functions and found that the best results were achieved with a fitness function based on an internal cross-validation (3-fold) coupled with a tiebreaker fitness that favours smaller KTree kernels. This investigation also found that the stability of KTree was improved by the use of a *Mercer filter*. Furthermore, a non-Mercer kernel does not define a distance metric as described in Section 2.1. The Mercer filter estimates the Mercer condition of a kernel by calculating the eigenvalues of the kernel matrix over the training data; if any negative eigenvalues are discovered, the kernel is marked as non-Mercer and is assigned the worst possible fitness, e.g. a cross-validation error of 100%. To reduce the computational cost when dealing with larger datasets, the kernel matrix is based on only a subset of the training data; this subset is randomly selected and the same subset is used in each kernel evaluation. For the experiments reported here, the kernel matrix was limited to a maximum size of 250x250.

4 Experimental Results

The goal of the experiments presented here is to determine if KTree can evolve kernel-based distance metrics that improve on the standard Euclidean distance when embedded in a k-NN regression algorithm. The next two sections describe experiments based on synthetic and real-world data and discuss the results.

4.1 Synthetic Dataset

A synthetic dataset, named *FeatureSpace*, was devised to comprise two predictor attributes and one dependent attribute, the value of which is to be predicted using k-NN regression. *FeatureSpace* is based on a specified feature mapping from a two-dimensional to a three-dimensional space. To create this dataset, 1000 two-dimensional points were randomly generated. The following feature space mapping was then applied to each point, x_i:

$$\begin{aligned} \phi_1(x_i) &= (x_{i1} - x_{i2})^2 \\ \phi_2(x_i) &= (x_{i1} + x_{i2} + 1)^3 \\ \phi_3(x_i) &= x_{i1} x_{i2} \end{aligned} \tag{5}$$

where x_{i1} is the value of the first attribute of sample x_i, x_{i2} is the value of the second attribute of sample x_i, and $\phi_p(x_i)$ is the value of the mapping of x_i along the

p-th axis in the new feature space. To generate the value of the dependent variable for each sample, y_i, for each sample, the following simple function is used:

$$y_i = f(x_i) = \phi_1(x_i) + \phi_2(x_i) + \phi_3(x_i) \quad (6)$$

Table 2 compares the performance of k-NN on the FeatureSpace dataset with different distance metrics. This experiment uses 200 samples of the FeatureSpace dataset for training and the remainder of the dataset as the test set. Table 2 shows the root relative squared error of prediction [9] achieved by k-NN with each distance metric on the test set. This table also shows the fitness of the final kernel selected by KTree and compares this with the 'fitness' of the two other distance metrics; the fitness of the Euclidean and FeatureSpace kernel distance is calculated using the same evaluation function as used by KTree, i.e. the average root relative squared error over a 3-fold cross-validation run on the training subset[3]. The results indicate that 3-NN with a distance metric based on the evolved KTree kernel improves on the performance of 3-NN with the standard Euclidean distance, both in terms of test error and fitness on the training data. Note that the higher errors over the training set may be due to the smaller training set used within the 3-fold fitness evaluation than that used for the test set.

Table 2. Results of 3-NN with different distance metrics on the FeatureSpace dataset. Both fitness and error values are the root relative squared error of prediction achieved by k-NN using each distance metric. Fitness is the 3-Fold CV root relative squared error over the training subset.

Distance Metric	**Training Set Fitness (3-Fold CV Error)**	**Test Set Error**
Euclidean	17.61%	14.51%
KTree	15.56%	11.15%
FeatureSpace Kernel	11.81%	6.54%

The final row of Table 2 shows the fitness and test error for 3-NN based on the FeatureSpace kernel. This kernel explicitly maps its two inputs according to the mapping defined in Equation 5 to generate two vectors of length 3, and then returns the dot-product of these two vectors. Using 3-NN with the FeatureSpace kernel is equivalent to operating 3-NN in the original three-dimensional feature space, in which the target function was defined. This result represents the best result that could be achieved with 3-NN. Note that the FeatureSpace kernel does not achieve 0% error as the training set does not provide 100% coverage of the target function; this was confirmed by calculation of the minimum theoretical error for 1-NN with the same training and test sets, which was found to be 5.16%. The results show that the performance of KTree on the test set is roughly half-way between that of the Euclidean distance and the FeatureSpace kernel distance. Despite the good performance of KTree relative to the benchmark of the Euclidean distance, the FeatureSpace kernel distance result shows that better kernels could possibly be found, e.g. by increasing population size or by increasing the maximum number of generations allowed.

[3] Since fitness is computed using an error measure, lower fitness values are better.

4.2 Real-world Datasets

To extend the results of KTree with k-NN on the synthetic dataset, a similar comparison of KTree with Euclidean distance was carried out, using a number of real-world regression datasets [10]. Table 3 shows the average error from a single 10-fold cross-validation run over 10 regression datasets; due to the relatively large size of the Abalone dataset (4177 instances) and the resultant significant increase in computation that would be required for KTree, a single subset of 250 instances was used for the training phase with the remainder being used as the test set. This table shows the results for 3-NN based on Euclidean distance and the results for 3-NN based on a KTree-evolved kernel distance. The lowest numerical test error on each dataset is highlighted in bold.

Table 3. k-NN regression 10-fold CV error rates (RMSEP): KTree-evolved distance vs. Euclidean distance. *The errors reported for this dataset are based on a single train/test split. Table also includes the number of samples and attributes of each dataset.

	No. Samples	**No. Attributes**	**Average Test Error**	
			Euclidean	**KTree**
Auto-MPG	398	8	2.86±0.47	**2.58±0.51**
CPU	209	7	0.04±0.04	**0.03±0.03**
Boston-Nox	506	14	0.04±0.01	**0.036±0.01**
Boston-Price	506	14	4.18±1.14	**3.45±1.11**
Octane	82	5	**0.61±0.16**	0.62±0.24
Deathrate	60	16	46.79±14.89	**46.63±10.49**
Bodyfat	252	15	2.87±0.36	**2.75±0.45**
Houseprice	117	7	212.97±62.89	**210.31±70.73**
Tecator	240	101	2.47±0.48	**2.35±0.38**
NO2	500	7	0.56±0.08	**0.55±0.08**
Abalone*	4177	11	2.69	**2.60**

For these experiments, the error is the root mean squared error of prediction (RMSEP). 3-NN using KTree achieves the lowest numerical test error on all datasets, except for the Octane dataset. A pairwise comparison over all datasets based on a Wilcoxon Signed Rank test [11] at a confidence level of 5% shows that 3-NN based on KTree outperforms the standard 3-NN. This demonstrates the ability of KTree to derive new distance metrics for a given data domain. In a similar analysis to that carried out for the synthetic dataset, the average fitness (based on 3-fold cross-validation RMSEP) of the Euclidean distance and KTree-evolved kernel distance was compared; the results are shown in Table 4. As was previously found with KTree for SVM classification, KTree is the clear winner in terms of the fitness of the distance metric it derives; this is also confirmed by the Wilcoxon Signed Rank test at a confidence level of 5%.

Overall, the results on both synthetic and real datasets demonstrate the effectiveness of the data-driven evolutionary approach of KTree when applied to a k-NN regression task. KTree is capable of evolving a kernel-based distance metric that is suited to a particular dataset. Distance metrics evolved by KTree could be used in library search applications, where the goal is to search for a list of the closest matches to a test sample.

Table 4. Average fitness (based on 3-fold CV RMSEP): KTree-evolved distance vs. Euclidean distance. *The fitness values reported for this dataset are based on a single train/test split.

	Average Fitness on Training Data			Average Fitness on Training Data	
	Euclidean	**KTree**		**Euclidean**	**KTree**
Auto-MPG	3.07±0.06	**2.57±0.07**	Bodyfat	3.44±0.12	**2.99±0.19**
CPU	0.06±0.01	**0.04±0.00**	Houseprice	221.38±13.42	**176.35±11.23**
Boston-Nox	0.05±0.00	**0.04±0.00**	Tecator	2.57±0.1	**2.44±0.1**
Boston-Price	5.12±0.21	**4.14±0.44**	NO2	0.57±0.01	**0.55±0.01**
Octane	0.71±0.04	**0.67±0.043**	Abalone*	2.48	**2.27**
Deathrate	48.82±3.27	**41.07±2.02**			

5 Related Research

Some previous work on the use of evolutionary algorithms with kernel-based learning has focussed on the optimisation of a single kernel, e.g. the RBF kernel for SVM classification [12]. In Lessmann *et al.* [13], a GA is used to optimise a set of parameters for five kernel types and the SVM C parameter, and is also used to determine how the result of each kernel is combined (addition or multiplication) to give the final kernel output. This approach is guaranteed to produce Mercer kernels, provided the Sigmoid kernel component setting does not break Mercer's condition. In comparison with KTree, however, the approach of Lessmann *et al.* is significantly restricted in the range of kernels that it can generate.

In more recent research, Gagne *et al.* [14] have proposed an approach for evolving kernels for a k-NN classifier. This approach is called the Evolutionary Kernel Machine (EKM). Although EKM bears some similarity to KTree in its use of GP to evolve a kernel, it differs considerably in a number of areas, including the kernel function representation and fitness measure used to evaluate candidate kernels. Their approach does not guard against non-Mercer kernels and the fitness function is based on k-NN specifically, i.e. the same fitness could not be used with SVMs, as is the case with the fitness function of KTree. Furthermore, EKM uses a co-evolutionary framework to evolve two subsets of the training dataset, a *fitness* and a *prototype* set, which are used in the fitness measure (see Gagne *et al.* for more details on this fitness measure). The authors do note that this competitive co-evolution can be problematic in that there is a danger of the fitness subset capturing noisy examples, thus resulting in a poor final model.

6 Conclusions

This paper has demonstrated the use of KTree to evolve a kernel-based distance metric for use in a k-NN regression algorithm. Experiments on both synthetic and real-world data showed that KTree is capable of evolving a distance metric that can improve on the widely used Euclidean distance. This represents a novel approach that facilitates the automatic discovery of a custom distance metric for a particular data domain. In building on previous work with KTree, these results also demonstrate that KTree can

be applied to different kernel methods and to different machine learning problems, i.e. classification and regression. One of the great advantages in the use of kernel methods is that they are easily adapted to work with different types of data; provided a kernel function can be defined for comparing two data objects, any kernel method can be applied to this data and a kernel-based distance metric may also be derived. Future work could investigate the use of KTree in structured data domains, such as the protein structure classification problems tackled by Wang & Scott [7]. The extension of this KTree research to tackle such problems may require the definition of new operators for the KTree kernel, which would allow it to manipulate structured data objects.

Acknowledgements

This first author's research has been funded by Enterprise Ireland's Basic Research Grant Programme. The second author acknowledges the support of a Marie Curie Transfer of Knowledge Fellowship of the European Communitys Sixth Framework Programme, Contract MTKD-CT-2005-029611. The authors are also grateful to the High Performance Computing Group at NUI Galway, funded under PRTLI I and III, for providing access to HPC facilities.

References

1. Karakoc, E., Cherkasov, A., Cenk Sahinalp, S.: Distance based algorithms for small biomolecule classification and structural similarity search. **22** (2006)
2. Yu, K., Ji, L., Zhang, X.: Kernel Nearest-Neighbour Algorithm. Neural Processing Letters **15** (2002) 147–156
3. Howley, T., Madden, M.G.: An evolutionary approach to automatic kernel construction. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN). (2006)
4. Cristianini, N., Shawe-Taylor, J.S.: An Introduction to Support Vector Machines. (2000)
5. Scholkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2002)
6. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research **2** (2002) 419–444
7. Wang, C., Scott, S.D.: New Kernels for Protein Structural Motif Discovery and Function Classification. In: Proc. of the 22nd International Conference on Machine Learning. (2005)
8. Cristianini, N., Shawe-Taylor, J.: Kernel Methods for Pattern Analysis. (2004)
9. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers (2000)
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases (1998)
11. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research **7** (2006) 1–30
12. Friedrichs, F., Igel, C.: Evolutionary Tuning of Multiple SVM Parameters. In: Proc. of the 12th European Symposium on Artificial Neural Network. (2004) 519–524
13. Lessmann, S., Stahlbock, R., Crone, S.: Genetically constructed kernels for support vector machines. In: Proc. of General Operations Research (GOR). (2005)
14. Gagne, C., Schoenauer, M., Sebag, M., Tomassini, M.: Genetic Programming for Kernel-based Learning with Co-evolving Subsets Selection. In: Parallel Problem Solving from Nature (PPSN IX). (2006)

A Study of Syntactic Information Retrieval

Chang Liu, Hui Wang, Sally Mcclean, Jun Liu, Shengli Wu

School of Computing and Mathematics, Faculty of Engineering, University of Ulster, Jordanstown, Shore Road, Newtownabbey, Northern Ireland, BT37 0QB
changliuuk@yahoo.co.uk, {h.wang, si.mcclean, j.liu, s.wu1}@ulster.ac.uk

Abstract. Natural language processing (NLP) techniques are believed to have the potential to aid information retrieval (IR) in terms of retrieval accuracy. In this paper we report a proof of concept study on a new approach to NLP-based IR that we proposed. Documents and queries are represented as syntactic parse trees, which are generated by a natural language parser. Based on this tree structured representation of documents and queries, the matching between a document and a query is executed on their tree representations, with tree comparison as the key operation. A classification experiment is designed to test if this approach is feasible. Experimental results show that this approach is promising and has the potential to significantly outperform the standard bag-of-word approach to information retrieval.

1 Introduction

Nowadays, natural language processing (NLP) techniques are increasingly applied to Information Retrieval (IR) to supplement the existing IR techniques. Most contributions of NLP to IR mainly concentrate on document representation and compound term matching strategies [4]. Researchers have noted that the simple term-based vector representation of document content is usually inadequate for accurate discrimination. For example, the Boolean IR model and the vector space model, usually called "bag-of-words" models of IR, can't tell the difference between the sentences "Mary is faster than John" and "John is faster than Mary". "Bag of words" IR models represent documents by a structure based only on a set of words and do not allow modeling of relationships between subsets of words. However, though much research has been done in order to represent documents by more accurate linguistically motivated content indicators [4], the matching strategy over documents and queries still cannot go beyond traditional statistical techniques that measure term co-occurrence characteristics and proximity in analyzing text structure.

In this paper, a new IR strategy is proposed with NLP techniques involved at the syntactic level. Within this proposed IR strategy, documents and the query are represented on the basis of a syntactic data structure of the natural language text – syntactic parse tree. Once this tree structured representation of documents and query is built, the matching between a document and a query is executed directly on the graph representation level, with the method of comparing trees as the key operation. It

is expected that the matching between documents and query over their tree structured representation will lead to reasonable improvement of the retrieval performance.

The organization of this paper is as follows: in the next section, we will give an overview of IR models as well as the NLP techniques used in IR which are relevant to our approach. In section 3, our syntactic IR approach is presented. In section 4, we describe a data classification experiment that is based on the matching of document titles extracted from the TREC dataset is designed and implemented to evaluate whether our approach (with different tree comparison methods) will outperform the traditional vector space IR model.

2 Background

2.1 Information Retrieval Model

Fundamentally, various IR systems are built upon on different IR models. The three classic models of IR are Boolean, vector space, and probabilistic IR models [3]. Variations of these models have also been developed over the years. Normally, IR models can be characterized into several categories [3]: document representation; query representation; a framework for modelling document and query representations and their relationships; and a ranking function which associates a real number to every document and query pairs [3]. This number is the measure of relevance and is determined by an incorporated matching strategy. Based on this number, the ordering/ranking among the documents with regard to the query can be defined.

The Boolean IR model is still the dominant model in the IR field. In the Boolean model, documents and queries are represented by a set of terms, which is the most basic and still remains the major way for representing documents. Strictly speaking, the Boolean model is not a “ranked retrieval model” such as the vector space model [1]. The Boolean model performs set operations over the documents and query representation (terms) and uses term co-occurrence characters to determine the retrieved list. On the other hand, the vector space IR model is an IR model that has a framework for ranking documents based on a similarity measure (partial matching) over documents and query. Once vectors have been computed for each document and query, the next step is to compute a numeric “similarity” between each pair. After sorting the resulting similarity scores, we can get the retrieved documents in a ranking list.

2.2 Natural Language Processing in Information Retrieval

The Natural Language Processing (NLP) approach to IR embraces all methods based on knowledge of the syntax and/or semantics of the natural language in which document text is written [2]. Such approaches attempt to address the structure and meaning of textual documents directly, instead of merely using statistical measures as

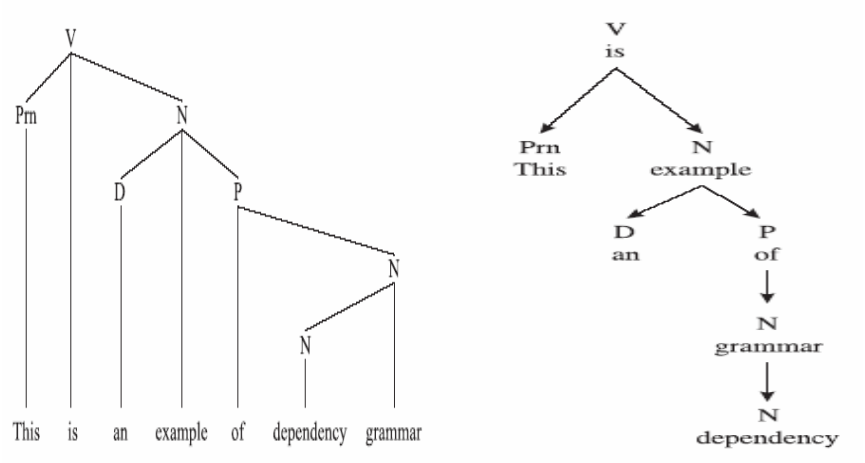
surrogates [2]. Most contributions of NLP to IR mainly concentrate on document representation and compound term matching strategies [4]. In response to the weakness of the bag of words model, much research work has been done to extract and make use of the syntactic structure information for representing documents and queries. The syntactic phrase is the logical representation view that straightforwardly comes into people's mind. The term syntactic phrase refers to "any set of words that satisfy certain syntactic relations or constitutes specified syntactic structures make up a phrase" [5]. As syntactic phrases capture actual linguistic relations between words rather than the single words (also better than the simple juxtaposition of words), they are regarded as a tool for increasing retrieval precision. For example, consider the query "river pollution". An IR system using natural language understanding would not retrieve a document saying "near to the river, air pollution is a major problem" with regard to the query that phrase indexed by "river pollution"; however, this document would be retrieved if the query is only indexed by single words "river" and "pollution", even though the document is obviously not about "river pollution". Syntactic phrases can be extracted from natural language sentences by NLP tools such as the constituent parser [11]. This parser can break a sentence into smaller syntactic phrases with respect to a given formal grammar, such as Noun Phrase, Verb Phrase, etc. The Noun Phrase is the syntactic phrase that is most used for phrase indexing in IR experiments [6][5][7].

2.3 Syntactic Parse Tree

Another data structure which captures the syntactic structure of the tokens in the textual sentence is the parse tree. There are two ways to describe the syntactic structure of natural language sentences in the form of "tree": phrase structure (PS-) trees and dependency (D-) trees [8]. A phrase structure (PS-) tree is generated by breaking up the sentence into constituents (phrases) according to a given formal grammar. Actually, the Noun Phrase or Verb Phrase are the constituents and are structured as the part of the phrase structure tree before they are extracted and used in the phrase based document representation. Figure 1 includes an example of the phrase structure (PS-) tree of the sentence "This is an example of dependency grammar". The dependency (D-) tree is an alternative way to describe syntactic structure of sentences in terms of dependencies between words. If two words are connected by a dependency relation, the dependent is generally the modifier, object, or complement; the head usually plays the larger role in determining the behaviour of the pair. A dependency tree is a set of links connecting heads to dependents which can easily form a tree. An example of the dependency tree of the same sentence "This is an example of dependency grammar" [9] is shown in Figure 1.

Mittendorfer and Winiworter [10] presented an algorithm for information retrieval that makes use of the syntactic structure of a query by exploiting its analyzed phrase structure trees. In their experiment, the query was split into sentences and parsed by the Link Grammar Parser (LGP) [11]. For each sentence, the LGP provides a set of possible phrase structure trees, which are combined into a parse lattice to be a graph

representation of the query. From this representation, an IR structure component computes a measure of connectedness c for each pair of words in the query. This quantity measures how strongly two words are bound together by the syntactic structure of the query and is taken as part of the formula for matching one document and the query.



Phrase Structure (PS-) Tree Dependency Tree

Fig. 1. Two ways of describing the syntactic structure of sentence "This is an example of dependency grammar"

3. Syntactic Information Retrieval – our approach

Within the trend of applying Natural Language Processing (NLP) techniques in Information Retrieval (IR), we aim to explore a different IR strategy in which the document and query representation are built on the basis of the syntactic parse trees and the matching function measures the similarity between document and query on their graph representation level (Figure 2). As we can see in the project [10], the query is represented as a combination of phrase structure trees parsed by Link Grammar Parser (LGP) [11]. In our project, instead of only parsing the query, we parse both sides of the query and documents by a selected parser (e.g. Minipar [12]) and create logical graph representations for both the document and query by exploring and making use of their analyzed syntactic parse trees. A framework will be built to transform the parse trees into an appropriate format for representing the documents and query as well as the later matching function. The format is a structured tree at this research stage. Given the parse tree based document and query representation, we propose a new matching strategy which measures the similarity between a document and a query directly on their graph (structured tree) level. A method of measuring the similarity between trees will be the major determinant in the overall matching strategy. For each document, a similarity score can be calculated from this matching

strategy and the retrieved documents will be ranked in the decreasing order of the similarity scores. With this different IR strategy based on parse trees and method for comparing trees, we expect it can improve the overall retrieval performance.

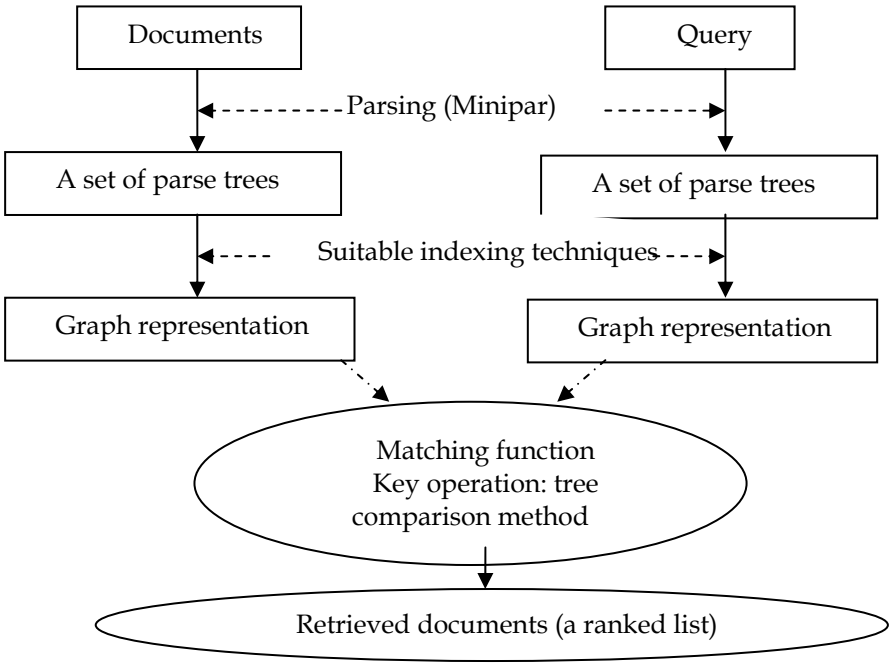


Fig. 2. Proposed syntactic information retrieval strategy

3.1 Documents and Query Representation Based on Parse Tree

In order to represent the documents and query on the basis of parse trees, the first task is to parse the natural language text from documents and query and then get a set of original parse trees. In our work, we plan to produce the dependency tree from the natural language documents and query text because we think dependency tree, which has fewer nodes than it peer phrase structure tree, is more clear and appropriate in the time consuming task such as IR. Minipar [12] is selected as the parser to perform this parsing task because the output of Minipar is a dependency tree. Minipar is a principle-based, broad-coverage parser for English [13] [12] and is available on Internet. It transforms a sentence as a network of nodes and links, where the nodes represent grammatical categories and the links represent types of syntactic dependency relationships. Minipar parses newspaper text at about 300 words per second on a Pentium-II 300 with 128MB memory [12]. An evaluation with the SUSANNE corpus shows that Minipar achieves about 88% precision and 80% recall with respect to dependency relationships [12].

As Minipar parse natural language text on the unit of sentence, a set of "raw" parse trees can be output after parsing a whole document text. The next step is to construct data structure of trees from the raw output of Minipar and then apply necessary techniques to these trees. The trees need to be modified into an appropriate graph representation format of the documents and queries for later matching on the graph level. This process is similar to the traditional document indexing procedure. A framework will be built for performing this task and it is one of the key tasks in this work.

3.2 Matching between the parse tree based documents and queries

Given a graph representation of the documents and queries on the basis of dependency trees, an IR matching strategy is proposed to measure the similarity between the documents and queries directly on their graph representations. The main operation in this matching strategy is the method of measuring similarity between trees. Two candidate methods have been selected so far and evaluated in the experiment. One is the Tree Edit Distance (TED) algorithm [14] and another is a recent algorithm, the All Common Subtrees (ACT) algorithm [18].

The TED is well known algorithm for comparing trees. Let T be an ordered rooted tree with vertex labels. Some edit operations on T can be defined as follows: (1) relabel: change the label of a node v in T; (2) delete: delete a non-root node v in T, then the children of v become the children of v's parent in T; (3) insert: inset a node v as a child of vv in T, making v the parent of a consecutive subsequence of the children of v [14]. Let T1 and T2 be two ordered, labeled trees, then each edit operation is associated with a cost. A cost C is the sum of the costs of a sequence of edit operations transforming T1 to T2. The TED between T1 and T2 is defined to be the minimum cost C of transforming T1 to T2, which can be used to measure the similarity score between T1 and T2. Several research groups have contributed to implementation algorithms for computing the TED. Shasha and Zhang [15]'s algorithm is the most cited one and is based on dynamic programming. Klein [16] and [17] also developed a TED algorithm based on dynamic programing which performance is better in terms of the worse case.

The ACT is a recent algorithm aimed at measuring the similarity between trees [18]. It advocates counting the number of all common subtrees as a way of comparing trees. The more the number of common subtrees of a pair of trees, the more one tree is similar to the other. ACT is theoretically inspired from the concept of Neighbourhood Counting Measure (NCM) [19]. NCM is a generic concept for measuring the similarity of two data points. It states that, give two data items (sets, vectors, sequences or trees), the number of their common neighbourhoods is an indication of the similarity between them. In ACT, the notion of a neighbourhood is interpreted as subtrees in terms of NCM. Consequently, the NCM becomes the counting of the number of all common subtrees. The authors of [18] have designed two ACT algorithms for computing the number of ACTs. One is a recursive algorithm based on

dynamic programming. It is easy to understand but less easy to control as is the case for all recursive algorithms [18]. This algorithm is polynomial in both time and space. The other algorithm is non-recursive with the time complexity O (|T1| ×|T2| × min (|T1|, |T2|) × max {depth (T1); depth (T2)}) and space complexity O (|T1| × |T2|).

By using tree comparison method as a key operation in the IR matching strategy, we aim to measure the similarity between documents and queries on their graph representations. For each document, a similarity score can be calculated from this matching strategy and the retrieved documents will be ranked in the decreasing order of the similarity scores.

4. Experimental evaluation

We designed and implemented an experiment to determine whether our syntactic IR approach can outperform the conventional vector space IR model, and also to determine which method of measuring similarity between trees perform better, TED or ACT. In the following discussion, ACTIR is an acronym for our syntactic IR approach using ACT as the matching function; TEDIR refers to our syntactic IR approach using TED; VSM is for the traditional vector space IR model. The methodology used in the experiment to evaluate the three IR approaches is data classification and we assume it has the same effect as the real IR experiment at this early research stage. By taking VSM as the benchmark, we can tell whether our syntactic IR approach will get higher accuracy, and at the same time, whether ACT performs better TED by comparing the accuracy of ACTIR and TEDIR.

Experimental set up

The data set used in this experiment is built upon a TREC data collection, *Text Research Collection Volume 2*, Revised March 1994, which includes material from the Wall Street Journal (1990, 1991, and 1992). The TREC data collections are produced by the TREC conference and each of the TREC data collections consists of a set of documents, a set of topics (queries), and a corresponding set of relevance judgments (right answers). In this experiment, based on the relevance judgments file, we selected 70 documents which are relevant to the topic 251, 70 documents which are relevant to topic 289 and 60 documents which are relevant to the topic 291. Therefore, an initial data set of 200 documents is created. Then, we extracted the title of each of the 200 documents and built a data set of 200 document titles, called T. In T, we assume that each title, which is a sentence or phrase, is able to represent the original document at this early research stage and is assigned a class label which is the topic the document is relevant to. Consequently, samples in T are classified into three groups: 70 samples belong to the group of topic 251; another 70 samples belong to the group of topic 289; the remaining 60 samples group the topic 291.

Once the data set T was constructed, we proceeded to build corresponding document representations according to the three IR approaches, ACTIR, TEDIR and VSM. For

ACTIR and TEDIR, we parsed the 200 document titles using Minipar [12] and obtained 200 corresponding dependency trees for representing the document titles. The implementation of Minipar we used in our experiment was extracted from GATE [22]. For VSM, we first built a simple terms dictionary from the 200 document titles. Then, 200 vectors were built for the 200 document titles based on the dictionary.

The next step was to design three different KNN classifiers [21] with regards to ACTIR, TEDIR and VSM. The major factor that distinguishes the three KNN classifiers is the method for calculating the distance between data samples. In the KNN classifiers of ACTIR, we use ACT [18] as the method to measure the distance between two dependency trees; For TEDIR, we use TED instead. Here, we use the TED function from Simpack [23] as our implementation of the TED. Cosine similarity [20] was used in the KNN classifier of VSM to measure the distance over vectors. In addition to the difference in distance metric, we also applied a small weighting scheme inspired by the Boolean IR model to the original distance values in the ACTIR and TEDIR classifiers. As the entire document title corpus is relevant to the three topics 251, 289 and 291, we defined a set of keywords for each topic. For example, the keyword of the topic 291 is “tax” and the keywords of the topic 289 are “Health”, “Medical” and “Hospital”. When the classifier of ACTIR or TEDIR calculates the distance between samples, a weight is put on the original distance value computed from ACT or TED if two samples both share the keywords from one topic.

Finally, we use *leave-one-out* [24] to estimate the three KNN classifiers in the task of classification. We take ACTIR as an example to explain the experiment procedure: in each iteration i within the *leave-one-out* method, applying the modified KNN classifier, we calculated the distance between the reserved test sample and the reaming 99 samples by using ACT [18]. If we assign K=3 in the first place, we chose 3 samples with the biggest 3 weighted ACT values among the 99. Then, the reserved test sample is grouped into the simple majority of the topic group of the three samples. If the classified topic group of the reserved test sample is the same as its original topic, we say the classifier for this iteration I, or for this reserved sample is correct. The accuracy is the overall number of correct classifications from the 200 iterations, divided by 200, the total number of samples in T. If the accuracy of ACTIR is higher than the one of VSM, we say our approach can outperform than VSM. If ACTIR has higher accuracy than TED in this experiment, we conclude ACT is a better method of measuring similarity between trees.

Experimental Results

The table below shows the experimental results when K is assigned from 1 to 8:

	ACTIR		TEDIR		VSM	
K	Correct Classifications	accuracy	Correct Classifications	accuracy	Correct Classifications	accuracy

1	173	86.5%	83	41.5%	172	86%
2	173	86.5%	83	41.5%	172	86%
3	173	86.5%	75	37.5%	166	83%
4	171	85.5%	76	38%	166	83%
5	171	85.5%	70	35%	168	84%
6	170	85%	70	35%	168	84%
7	163	81.5%	72	36%	167	83.5%
8	162	81%	72	36%	165	82.5%

Table 1. Comparison of Experiment Result

Analysis of Experiment

We can see from Table 1 that ACTIR has slightly higher accuracy values than VSM, while the accuracy values of TEDIR are significantly lower. For ACTIR and TEDIR, All Common Subtree (ACT) and Tree Edit Distance (TED) is taken as the key operation of matching among documents titles, while the small weighting scheme inspired by the Boolean IR model has the positive effect on increasing the accuracy. Results of this study show that our approach can outperform the standard bag-of-word approach, although we have not post-processed the parse trees at all. We believe that if the parse trees are processed properly the performance of our approach could be higher. It is our opinion that this approach is promising and has great potential.

5. Conclusion and Future Work

In this paper we present a proof of concept study of a new NLP-based approach to IR. In this approach documents and queries are represented as syntactic parse trees, which are generated by a language parser, and document ranking is achieved by computing the similarity between a query tree and a document tree. In this study we used Minipar as the language parser, which produces dependency trees (a form of parse tree) for documents. Additionally we considered two tree similarity measures, the well know *tree edit distance (TED)* and the new all common subtrees (ACT) similarity recently developed in our group [18]. We used an implementation of the vector space IR model as our benchmark. We experimented on a selection of TREC data and we found that (1) ACT is much better than TED on this task (2) our syntactic IR model can outperform the vector space model. These findings give us confidence in our syntactic IR model.

We plan to improve our implementation of the syntactic IR model in the following ways:

1. Prune the parse trees so that the retrieval accuracy is optimal

2. Index the parse trees so that the model can be scaled to large collections of documents.

References:

1. C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval. Cambridge UP, 2007, Draft.
2. Ed Greengrass, Information Retrieval: A Survey, 2000.
3. R. Baeza-Yates, B. Ribeiro-Neto, Addison-Wesley, Modern Information Retrieval. 1999.
4. Tomek Strzalkowski, Natural Language Information Retrieval, GE Corporate Research & Development, Schenectady, NY, USA
5. Mandar Mitra, Chris Buckley, Amit Singhal, Claire Cardie, An Analysis of Statistical and Syntactic Phrases, Proceedings of RIAO-97, 5th International Conference ``Recherche d'Information Assistee par Ordinateur",1997
6. David A. Evans, Chengxiang Zhai, Noun-Phrase Analysis in Unrestricted Text for Information Retrieval, Proceedings of the ACL-96, 34th Annual Meeting of the Association for Computational Linguistics, 1996
7. Arampatzis, A. T., Tsoris, T., Koster, C. H., and Van Der Weide, T. P. 1998. Phrase-based information retrieval. Inf. Process. Manage. 34, 6 (Nov. 1998), 693-707.
8. Melcuk, I. A. Dependency Syntax: Theory and Practice, State University of New York Press, 1988.
9. Michael A. Covington, A Fundamental Algorithm for Dependency Parsing, 2000
10. Mittendorfer, M. and Winiwarter, W. 2002. Exploiting syntactic analysis of queries for information retrieval. Data Knowl. Eng. 42, 3 (Sep. 2002), 315-325.
11. Sleator, D., Temperley, D., Parsing English with a Link Grammar. In: Proc. 3rd Intl. Workshop on Parsing Technologies, 1993.
12. Minipar Home Page, see http://www.cs.ualberta.ca/~lindek/minipar.htm
13. Lin, D., Principle-based parsing without overgeneration, In 31th Annual Meeting of the Association for Computational Linguistics (ACL 1993), 112-120, Columbus, 1993.
14. Bille, P. 2005. A survey on tree edit distance and related problems. Theor. Comput. Sci. 337, 1-3 (Jun. 2005), 217-239.
15. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing, 18:1245{1262, December 1989.
16. P.N. Klein. Computing the edit-distance between unrooted ordered trees.In Proceedings of the 6th annual European Symposium on Algorithms (ESA) 1998., pages 91–102. Springer-Verlag, 1998.
17. Erik Demaine, Shay Mozes, Benjamin Rossman , Oren Weimann, "An Optimal Decomposition Algorithm for Tree Edit Distance". In Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP 2007). To appear.
18. ACT: All Common Subtrees. Technical Report, University of Ulster, Submitted, 2007.
19. H. Wang. Nearest neighbors by neighborhood counting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(6):942-953, 2006.
20. Salton, G., McGill, M.J. Introduction to Modern Information Retrieval, McGraw Hill Publishing Company, New York, 1983.
21. T. Mitchel, Machine Learning. USA: McGraw-Hill Education, 1997.
22. GATE Home Page, see http://gate.ac.uk/
23. Simpack Project Page, see http://www.ifi.unizh.ch/ddis/simpack.html
24. J. Han and M. Kamber, Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

Rule-Based Khmer Part-of-Speech Tagging with Generalized Unknown Word Handling

Chenda NOU and Wataru KAMEYAMA

Graduate School of Global Information and Telecommunication Studies
WASEDA University
1011 Nishi-Tomida, Honjo-shi, Saitama-ken, 367-0035, Japan
chendanou@fuji.waseda.jp, wataru@waseda.jp

Abstract. Part-of-speech tagging is a fundamental step in natural language processing system. It is required to achieve high accuracy to be used in other high-level language processing works. In this paper, we propose some modifications on transformation-based approach to obtain higher accuracy than our previous work. In addition, to handle the unknown word problems, we introduce an automatic rule generation method to extract feature rules from training corpus. The tagger achieves 94.83% and 90.70% of tagging accuracy on the training and test data respectively.

Keywords: Khmer POS tagging, transformation-based tagging, corpus-based, automatic rule generation, unknown word POS guessing

1 Introduction

Part-of-speech (POS) tagging makes sentences easier to parse by computer, and is therefore a preprocessing step frequently required in natural language processing system.

Two main approaches for this task are stochastic approach [4], [8] and rule-based [1], [5], [6] approach. Stochastic approaches had often been preferred to rule-based because of their automatic training capabilities. This was the case for POS tagging until Brill introduced a rule-based tagger [1] by inferring rules from a training corpus and achieved as high accuracy as the stochastic approaches. Anyway, Brill's approach, the transformation-based approach, provides the ease of adjusting to new languages such as: i) templates of the rules can be defined based on sentence structure and characteristics of the languages, ii) a small set of human readable rules provide the ease of finding problems that affect the tagging accuracy and implementing the improvement.

In our previous work [3], we have proposed the initiative POS tagger for Khmer by using transformation-based approach. The work achieves 96.1% and 95.12% of tagging accuracy on the training and test data respectively without considering unknown word problems.

In order that the result of tagging will be useful for high-level language processing researches, high accuracy with considering unknown word problems is required.

Therefore in this paper, we propose some modifications on transformation-based approach to reduce the tagging error. Furthermore, we present the automatic feature rule extraction and ranking methods for the task of unknown word POS guessing.

2 Characteristics of Khmer Language and Background

2.1 Characteristics of Khmer Language

Khmer, one of the prominent Austro-Asiatic languages, has different characteristics from English language. The list below shows the characteristics of Khmer that must be considered for POS tagset and tagging system design, especially for solving unknown words.

- It is written continuously without delimiter and no capitalized structure. The collocation and the boundary between compounds and phrases are fuzzy.
- Basically, it is a subject-verb-object language but has relatively free word order as compared to English. For instance, an adjective is placed directly after its subject without linking verb.
- It is devoid of inflection in either nouns or verbs.
- Nouns and verbs are not gender-sensitive.
- A word is a combination of consonants, vowels, subscripts (sub-consonants), and/or diacritics. A sub-consonant, a vowel, and a diacritic cannot stand alone without a consonant. Thus, the length of a word is calculated by the number of clusters not the number of characters. A cluster must have one consonant, and/or subscripts, and/or vowel, and/or diacritic. e.g. ក្នុង "In" has two clusters: the first cluster "ក្នុ" is a combination of the consonant "ក" , the subscript "◌្", and the vowel "◌ុ", while the second cluster has only one consonant "ង".
- Different ways of writing: some words with the same pronunciation and the same meaning, but can be written in various ways. e.g. អោយ, ឲ្យ, ឱ្យ [*oy*] "Give, Let"
- The prefix can give more accurate information about the word than the suffix does.
- A word can be a combination of many words.
  e.g. ទ្បាន-ដឹក-ទំនិញ [noun+verb+noun] "Truck"
- The same pattern of word has many different POS. For example, POS of words which are a combination of "verb+ noun" can be adjective, adverb, noun, or verb.
- The longer prefix or suffix can give more accurate information about POS of the word than the shorter ones.

2.2 Khmer Unknown Words

Unknown words are words which don't exist in the lexicon. In general, there is no new word invented in closed word classes such as preposition, determiners or

conjunctions. Therefore, we consider unknown words fall under 9 open word classes: noun, adjective, verb, adverb, acronym, proper name, participle, ordinal number, and exclamation word. Khmer unknown words are categorized into the following 6 types:

1. acronym, a shorten form of long names or phrases,
2. transformed words, words that are transformed from other words by using prefixes or suffixes. e.g. ការសំរេចចិត្ត "decision" is a combination of ការ and សំរេចចិត្ត "to decide". ការ is a prefix used to change the word from verb to noun,
3. proper nouns include names of people, locations, and organizations,
4. compound words consist of more than one stem, e.g. ក្រោក-ឡើង "*get up, or stand up*", ទី-២ "*2nd, second*", បាន-ល្អ "*well*",
5. reduplicating words to make plural noun or emphasize the meaning of the word by using the "ៗ" symbol, e.g. ក្មេង-ៗ [*kmeng kmeng*] "*kids*, where *ក្មេង* [*kmeng*] means *kid* or *young*", ដដែល-ៗ [*dordel dordel*] "*always the same*, where ដដែល [*dordel*] means *the same*" ,
6. new words or loan words, currently don't exist in our wordlist, e.g. កំព្យូទ័រ "*computer*", ប៊ឹស "*bus*".

2.3 Background

In our previous work [3], we have proposed the first Khmer POS tagger by using transformation-based approach. Due to the lack of resources in Khmer, we have setup some resources such as tagset, tagged corpus, and lexicon.

In this paper, we use the same tagset, corpus, and lexicon in our previous work [3]. The tagset contains 27 tags which have been shown to be useful in other natural language processing systems. The lexicon consists of 32,000 words which are tagged with the defined tagset. The corpus includes 37,452 words (1102 sentences) retrieved from Kohsantepheap daily [7], one of the famous newspapers in Cambodia.

In order to avoid the domain-based results, in this paper, we add some more data retrieved from various domains such as Khmer legends, letters, etc. into the testing corpus.

To segment a sentence into words, we use segmentation API provided by PAN [2]. In our previous work [3], we have added some modules to reduce the segmentation errors. In this paper, we conduct experiments on the segmented corpus, thus errors caused by segmentation are not considered.

3 Khmer POS Tagging System

Fig. 1 shows the architecture of the tagging system. First, the tagger looks into the lexicon and defines the most frequent tag to each word. If the word is not found in the lexicon, it is passed to the guesser. The guesser will define the most likely tag to the unknown word. Finally in the transformation process, the tagger may change some

given tags if the context of the words match to criteria of any transformation rules. The transformation rules can be extracted automatically from the training corpus by rule learning process.

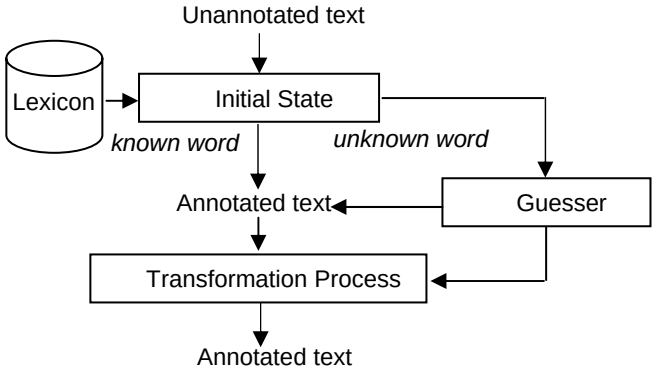


Fig. 1. Architecture of the system

3.1 Rule Learning

We adopt Brill's algorithm [1] to automatically learn the error-correcting rules from the annotated corpus. The procedure of the learning algorithm is summarized as follows:

```
a. Initial tagging
b. Automatic rules generation based on predefined
   templates
c. Getting the rule which causes the best improvement
d. Applying the best improvement rule on the training
   corpus
e. Saving the best improvement rule
f. Repeating step b to e, until the best improvement
   rule cannot meet the predefined threshold
```

Fig. 2. Learning Algorithm

We have defined 32 transformation templates in [3] based on the study of tagging errors obtained from the initial tagging phase and the characteristics of Khmer language. There are two categories of the templates:

Category 1: *Change tag from A to B if conditions are fulfilled*

Category 2: *Change tag from A to B if current word is W and conditions are fulfilled*

The following are some examples of the templates:

Change tag A to B when:

```
Previous tag is X and next tag is Y
Current word is W and any of two next tags is X
```

3.2 Transformation Process

The transformation process is a process to apply the learned rules obtained from the learning process to reduce errors caused by the initial tagging. The rules are to be applied in the form of the two categories templates.

Two types of unexpected errors may occur after applying a transformation rule: 1) the new tag (destination tag of the rule) is one of tags associated to the word, but it is not a correct tag in a context, 2) the new tag is not one of tags associated to the word, and it is also not a correct tag in a context. The error 1) and 2) may appear when applying the rules in category 1. The rules in category 2 have been restricted on a specific word, thus only error 1) may appear.

As an example of the second type error, a rule *"change verb to preposition if previous tag is verb"* is learned and 3 input sentences with initial tagging are:

i) គាត់[PN]ដើរ[V]*ទៅ/V*សាលា[N]។

ii) គាត់[PN]រត់[V]*មក/V*ផ្ទះ[N]។

ii) គាត់[PN]ចង់[V]*ទិញ/V*ឡាន[N]។

PN: Pronoun, V: Verb, N: Noun

The condition of the rule is matched to all the example sentences on the italic words. But, the rule is applicable only to i) and ii), because in these cases, ទៅ and មក function as a preposition "to". If they are not preceded by a verb, they function as verbs which ទៅ means "go" and មក means "come". The rule is not applicable for iii). Because ទិញ "buy" is always a verb, and cannot be a preposition. Thus, if the rule is applied directly when the contextual conditions are fulfilled, the rule will change the tag of ទិញ from the correct tag to the wrong tag.

We can easily avoid this kind of errors by looking at the tags information associated to the word in the wordlist. In the wordlist, the word ទិញ contains only one tag, verb.

Due to this problem, we propose a change on applying rule algorithm: *a rule is to be applied to only any word that is in the context of the rule and the destination tag is one of the tags associated to the word.* If the destination tag is not in the tag set of the word, we can know that it is an incorrect tag although the word is in the context of the transformation rule.

However, not all proper noun and participle tags are attached to their associated words in our wordlist. Therefore, if the destination tag of a rule is a proper noun, the criterion is changed to: *the transformation rule is to be applied, if word is in the context of transformation rule, and one of the tags associated to the word in the lexicon is {Noun, Adjective, Verb, Adverb, or Numeric}*. This is because in Khmer,

only the word that belongs to noun, adjective, verb, adverb, or numeric class can be a proper name.

If the destination tag of a rule is a participle, then the criterion is changed to: *the transformation rule is to be applied, if word is in the context of transformation rule, and one of the tags associated to the word in the lexicon is verb.*

The algorithm of the method is roughly described as follows:

```
ApplyingRule (rule)
 {
      Q= all possible tags of the current word
      P=  {noun, verb, adj, adv, numeric}
          tag=destination tag of the rule
      if  (current word is in context of the rule) AND
          ( ( tag ∈ Q) OR
           (tag=ProperNoun AND  ( P ∩ Q) !={∅}   ) OR
           (tag=Participle AND (Verb ∈ Q) )
          )
             Apply rule
 }
```

Fig. 3. Applying rule algorithm

Of course, the proposed method is not a complete solution to eliminate all errors created by applying transformation rules, but it can reduce many errors caused by the change from a correct tag to an incorrect one.

The same problem also happens in applying rules in the learning process. Thus, this algorithm is also applied to step *d (Applying the best improvement rule on the training corpus)* in Fig.2.

4 Unknown Word POS Prediction

Rule-based approach is one of the effective approaches to predict the POS of unknown words. This approach predicts the most likely tag for unknown words based on the internal structure of words such as suffix, prefix, word-form (e.g. noun + verb, adjective + noun, etc.).

Three main problems of this approach are: how to create the rules, how to rank the rules, and how to apply those rules effectively. It is quite difficult and time consuming to write down all possible rules. Thus, we propose a learning algorithm to learn rules automatically from corpus. Then, the rules are ranked based on the ranking method by considering the characteristics of the language. Finally, the rules are applied based on the precision of each rule type.

4.1 Rule Generation

To generate all feature rules for predicting the POS of the unknown words, we propose a simple algorithm to learn the rules automatically based on our pre-defined templates. The process of the algorithm is roughly described as follows:

```
for t=1 to No. of Templates
    for each open-class word in training data
       Create all possible feature rules based on
         Templates[t]
    Calculate the frequency of each rule
    Get only rules with frequency >=Minimum frequency
    Order the rules based on the ranking method
    Save rules to rule list[t]
```

Fig. 4. Feature rules generation algorithm

In general, features that rarely appear in the training data are statistically unreliable and often decrease the performance of the system. Thus, only the rules with frequency larger than or equal to the minimum frequency are saved to the rule list. The minimum frequency should be set in accordance with the size of the training data. The value of the minimum frequency can affect the accuracy and performance of the guesser. The low minimum frequency may result in some unreliable rules. Anyway, the performance of guessing is slow as the guesser has to check a lot of rules. If the minimum frequency is too high, some valid rules might not be saved but the performance of the guessing is faster. Due to the small size of the training data, we set the minimum frequency to 1 in this paper. The accuracy of POS guessing by various frequencies is shown in Section 5.

The allowable transformation templates 1 to 5 are listed below:

Attach tag X to an unknown word if:

1. If the word contains any symbol {., -, /, _ } and/ or numeric
2. If first (last) (1,2,3) clusters of the word are *w* and other clusters results in a word whose tag is *x*
3. If first (1,2,3) clusters results in a word whose tag is *x,* and next (1,2,3) clusters results in a word whose tag is *y*, and the other clusters results in word whose tag is *z*
4. If first (1,2,3) clusters results in a word whose tag is *x* and other clusters results in a word whose tag is *y*
5. If first (last) (1,2,3) clusters of the word are *w* and results in a word

w is a part of a word, suffix, or prefix. *x*, *y*, and *z* are one of the 27 defined tags.

4.2 Rule Ranking Method

The rules in each template are ranked by frequency in descending order. Based on the characteristics of Khmer, all the prefix rules have higher priority than the suffix rules. The 3-cluster rules have higher priority than the 2-cluster rules, and sequentially.

4.3 Applying Rule

The order in which the rules in each set are applied is crucial for dealing with ambiguous cases. The problem can be solved, if the highest reliable rules are applied first. Thus, we define the order of applying rule based on precision of each template in the task of guessing the unknown words. We apply all rules in each template to predict the POS of 1,872 unknown words in the training data. The precisions of guessing unknown words by rules in each template are shown in Table 1.

As shown in Table 1, the rules in the first template gain the highest precision. Therefore, the rules in template 1 should be applied first, and sequentially followed by rules in template 2, 3, 4, and 5.

Table 1. Precision of guessing POS of unknown words by rules in each template

Template	1	2	3	4	5
Precision%	100	94.95	91.98	86.75	85.90

5 Experiments

5.1 Experiments Setup

We divide the corpus into two: 32,088 words (47% are ambiguous words) for training and 8,969 (48.4% are ambiguous words) for testing. All the training data are from only one source, Kohsantepheap newspaper [7]. The test data are retrieved from two different sources: 60% is from the same newspaper as the training data, hereafter Test Data 1, and other 40% are from various domains, hereafter Test Data 2.

For the experiments on handling of unknown word, we delete 2,681 open class words which appear only once in the training and test corpus from the wordlist. Thus, 5.83 % of words (1,872 words) in the training corpus and 9% of words (809 words) in the test corpus are unknown words. The distribution of unknown words in each word class is shown in Table 2.

. **Table 2.** Distribution of unknown words in each word class

Tag	**Description**	**Training data**	**Test data**
N	Noun	1,143	390
PN	Proper noun	23	15
V	Verb	385	203
Adj	Adjective	130	92
Adv	Adverb	141	81
ORD	Ordinal number, a combination of characters and number. e.g. ๕๕5	32	22
AC	Acronym	10	4
PP	Participle verb	7	0
EX	Exclamation word	1	2
	Total	*1,872*	*809*

All the experimental results are calculated as recall(R), precision (P), and F-score (F) which are defined as follows:

R= Total relevant records retrieved / All relevant records

P= Total relevant records retrieved / All records retrieved

F= (2* R * P) / (R+P)

5.2 Results and Discussions

To show the effectiveness of the modification on applying rule algorithms, we conduct experiments by using transformation-based approach: i) without any modification, ii) modify the applying rule algorithms. The tagging results (without unknown words) in Table 3 show that the proposed modifications doesn't improve much accuracy on the training data, but it can effectively improve the accuracy of test data, especially the Test Data 2 which is from different domain with the training data.

Table 3. Accuracy of Brill's technique and proposed technique

Recall	Training Data	Test Data 1	Test Data 2	Test Data 1+ Test Data 2
Brill's	96.10%	95.12%	93.58%	94.35%
Proposed	96.16%	95.64%	94.56%	95.10%

Table 4 shows the accuracy of unknown word POS prediction by rule-based approach with different minimum frequencies (MF). The approach achieves highest recall when the MF is set to one. But, the test data obtains the highest precision when MF is set to 5, and the precision of MF 5 is also higher than the precision of MF 3 in the training data. These results illustrate that with the high MF, we can obtain more reliable rules. However, we may also lose some valid rules which have lower frequencies.

Table 4. Accuracy of unknown word POS guessing by different minimum frequencies

Minimum Frequency	No. of rules obtain	Training Data			Test Data 1 +Test Data 2		
		R%	P%	F%	R%	P%	F%
1	9629	87.66	93.87	90.65	64.40	71.66	67.83
3	961	67.30	82.46	74.11	55.62	74.62	63.72
5	412	62.12	84.15	71.47	51.42	77.61	61.86

We combine the proposed tagging technique with unknown word handling method to tag the training and test data in which 5.8% and 9% of words respectively are unknown. Table 5 illustrates the overall tagging accuracy of both data. The recall of unknown word on the test data has been slightly improved after applying the transformation process.

Table 5. Overall tagging accuracy by proposed technique with rule-based unknown word POS prediction

Recall	Known word	Unknown word	Overall
Training data	95.27%	87.66%	94.83%
Test Data 1 + Test Data 2	93.30%	64.41%	90.70%

6 Conclusion

Based on the study of tagging errors obtain in our previous work [3] and specific characteristics of Khmer language, we have proposed some modifications on applying rule algorithm with the limitation that: the transformation rule is to be applied only when the word is in the context of the rule and destination tag of the rule is one of the tags associated to the word. The proposed technique has been proven to be effective in reducing tagging errors especially when tagging unseen texts from various domains.

To handle the unknown word problems, we propose the automatic rule generation method to extract the feature rules from the training data. The approach achieves very encouraging results. However, the rule-based approach makes use only of the internal structure of the words. It seems to neglect the advantages of contextual information such as surround tags and words information which also can contribute to this work.

Therefore to obtain a higher accuracy in predicting POS of unknown words, we are going to combine the rule-based and statistical approach which makes use of the contextual information for this task. The combination approach combines the strengths of both approaches and is expected to achieve higher accuracy than any single approach.

References

1. Eric Brill: Transformation-based error-driven learning and natural language processing: a case study on part of speech tagging. Computational Linguistics, vol. 21, n.24 (1995)
2. CHEA Sok Huor, TOP Rithy, ROS Pich Hemy: Detection and correction of homophonous error word for Khmer language. Ref. No. PANL10n/Admn/RR/.
3. Chenda NOU, and Wataru KAMEYAMA: Transformation-based Khmer part-of-speech tagger. In The 2007 International Conference on Artificial Intelligence, WORLDCOMP' 07. Las Vegas, US. Vol.1, pp. 581-587 (2007).
4. Cutting Doug, Julian Kupiec, Jan Pederson, and Penelope Sibun: A Practical Part-of-Speech Tagger. In Proceedings of the Third Conference on Applied Natural Language Processing, ACL (1992)
5. Green, B. and Rubin, G.: Automated Grammatical Tagging of English. Department of Linguistics, Brown University (1971)
6. Klein, S. and Simmons, R.F.: A Computational Approach to Grammatical Coding of English Words. *JACM* 10: 334-47 (1963)
7. Kohsantepheap Newspaper. July, 2006. Available: http://www.kohsantepheapdaily.com.kh
8. Thorsten Brants : TnT – A Statistical Part-of-Speech tagger. In Proceedings of the 6th Conference on Applied Natural Language Processing, pages 224-231 (2000)

Sticking with a Winning Team: Better Neighbour Selection for Conversational Collaborative Recommendation*

Rachael Rafter, Lorcan Coyle, Paddy Nixon, and Barry Smyth

Adaptive Information Cluster,
School of Computer Science and Informatics,
UCD Dublin, Ireland
firstname.lastname@ucd.ie

Abstract. Conversational recommender systems have recently emerged as useful alternative strategies to their single-shot counterpart, especially given their ability to expose a user's *current* preferences. These systems use conversational feedback to hone in on the most suitable item for recommendation by improving the mechanism that finds useful collaborators. We propose a novel architecture for performing recommendation that incorporates information about the individual performance of neighbours during a recommendation session, into the neighbour retrieval mechanism. We present our architecture and a set of preliminary evaluation results that suggest there is some merit to our approach. We examine these results and discuss what they mean for future research.

1 Introduction

Traditionally, collaborative recommender systems are based on a *single-shot* model of recommendation where a single set of recommendations is generated based on a user's (past) stored preferences [1]. Such systems assume that users have stable preferences which can be represented with a static user profile that grows over time. However, content-based recommender system research has begun to look towards more *conversational* models of recommendation, where the user is actively engaged in directing search at recommendation time [2–4].

Previously we have proposed adopting a similar model for collaborative recommendation [5] where recommendations are made in a cyclical process, and the user provides *preference-based feedback* [6] on the suitability of the items recommended after each *cycle*. This feedback is used to model the user's current *short-term* preferences. This has been motivated by our belief that a user's *current* preferences are influenced by her current mood, and moreover that they may deviate from her regular movie preferences. Hayes and Cunningham [7] have similarly pointed out that Collaborative Recommendation can lack sensitivity to a user's current interests and may cause frustration and distrust. They

* This work is partially supported by Science Foundation Ireland under grant number 04/RPI/1544 "Secure and predictable pervasive computing".

propose to boost collaborative recommendations with a more knowledge-heavy approach than ours, that incorporates context or task driven information into the recommendation process. Although it could be argued that this conversational approach imposes a burden on the user to provide feedback, we would maintain that it can also help the user to explore the information space in a more useful manner.

Recent results from our research [5] have already indicated that this conversational model of recommendation has advantages over its static single-shot counterpart. Bridge and Kelly [8], went further and incorporated diversity into the recommendation process which improved results. In this paper, we focus on enhancing the selection of *nearest neighbours* (NNs) which is crucial to any recommender system. We propose that the feedback provided by the user during a session, can not only be used to model the user's current preferences, but also to identify those neighbours that are performing best as recommendation partners for the target user, given her current preferences.

Our contributions in this article are twofold. We propose a novel component called the *Sticky Layer* that controls the promotion and demotion of neighbours based on their performance to date during a recommendation session; this layer performs a type of "recommendation priming" that is guided by the recommendation session. We also propose an extension of Hayes *et al.*'s [9] Case Retrieval Net (CRN) implementation of k-NN retrieval for Automated Collaborative Filtering, which takes the effectiveness of an individual user's past recommendations in the current recommendation session into account, when retrieving new recommendations. The CRN is supported by the Sticky Layer.

The rest of this paper is organised as follows: In Section 2 we provide some background. In Section 3 we discuss the novel aspects of our system. In Section 4 we discuss some preliminary evaluation results, and in Section 5 we conclude and propose future research. We should note here that the work presented in this article is still not fully mature, and our evaluation reflects ongoing work. However, these preliminary results present some interesting ideas that have emerged from a considerable amount of research. We try to raise some open questions, which we hope will stimulate interesting future research.

2 Background

In this section we discuss collaborative recommendation and conversational recommendation in brief, and then we provide some background on Case Retrieval Nets and how they can be used for collaborative recommendation.

2.1 Collaborative Recommendation

Single-shot Collaborative Recommendation (SS-CR) is a *content-free* recommendation strategy based on the premise that similar users like similar items. Recommendations are compiled for the *target user* based on the profiles of her *nearest neighbours* (NNs). Importantly, the profile only contains information about the

user's *long-term* preferences, and ignores any *short-term* preference differences that the user may have. Therefore user profiles where *both* short-term and long-term preferences of the user are represented, have been proposed, e.g. [10, 11].

In Conversational Collaborative Recommendation (C-CR) [5] the standard long-term profile from the single-shot model, is complemented by a short-term profile which models the user's more current (and possibly transient) preferences. Here, *cycles* of k item recommendations are made to the user; after each cycle she is asked to indicate which recommendation would be most suitable, or else indicate that none are suitable (by selecting one recommendation, or by rejecting all of them, respectively). This feedback is then incorporated into her short-term profile and used to alter the similarity retrieval process. This process is repeated, with new items being recommended to the user each time based on the updated profile, until the user finds an item with which she is satisfied. We refer to the entire set of cycles resulting in a satisfactory recommendation as a *session*.

2.2 CRNs for Collaborative Recommendation

Hayes *et al.* [9] pointed out the similarities between Collaborative Recommendation and Case-Based Reasoning (CBR). Core to the CBR methodology is the use of a similarity retrieval mechanism for finding the k nearest neighbours (k-NN) to the current problem (or target) case. Case Retrieval Nets (CRNs) have been used to implement k-NN similarity retrieval due to their ability to efficiently and flexibly retrieve similar cases [12]. CRNs outperform the standard k-NN approach in domains where there is feature-value redundancy or domains with many missing feature-values.

Hayes *et al.* proposed that CRNs could be used to perform collaborative recommendation (or Automated Collaborative Filtering) by treating a user profile like a case and the user recommendations on individual items as case features [9]. These cases are usually sparse, playing to the strengths of the CRN, and allowing for efficient retrieval of the most similar users to the current target case. Information from the NNs that emerge can then be used to make a recommendation to the target user. This approach had two major limitations:

- Features are represented as item recommendation pairs (IEs), whereby the CRN typically contains more than one IE for each item, (one IE for every distinct recommendation score for that item).
- Similarity measures need to be explicitly defined between IEs. This typically means that a domain expert must specify how any two features relate to each other. This implies a knowledge-intensive approach.

To overcome these limitations, we take inspiration from the work of Delany *et al.* [13] on using CBR in the spam filtering domain. They used CRNs for retrieving similar e-mails to classify a target e-mail. The spam-filtering domain is interesting for CRN development since all features are binary — spam features just represent words in an e-mail, each feature is true if the word it represents is present in an email, and false otherwise. With binary features, the CRN can

be connected in such a way that calculating similarity is not necessary (the assumption is that similarities between features are always zero).

Given that the use of the CRN structure comes from the CBR field it should be pointed out that the process of incorporating user feedback into a similarity retrieval process as we do has an analogue there. Richter [14] has proposed that CBR has four distinct knowledge containers (the case-base, vocabulary used, similarity measure, and solution transformation) and that it may be advantageous to move knowledge between them. The CBR analogue to our work would be a system that moved (or incorporated) knowledge from the solution transformation to the similarity measure.

3 Implementation

Recommendations are generated using user profiles retrieved from a novel representation of a CRN called a Collaborative Case Retrieval Net (*CCRN*). Our CCRNs are an adaptation of the CRN similarity retrieval implementation from the Fionn CBR framework [15].

Traditional CRNs are made up of the following components (the differences between CRNs and CCRNs are also outlined), which are illustrated in Fig. 1:

- Nodes represent stored CBR cases (traditional CRNs call these *case nodes*). CCRN nodes represent individual user profiles.
- Traditionally, Information Entities (IEs) represent feature-value pairs within cases. CCRN IEs represent items that are available for recommendation.
- Relevance Arcs link case nodes with the IEs that represent them. Typically they have weights that capture the importance of the IE to the connecting node. CCRNs use these to capture both the items that were preferred by the user as part of their permanent profile and items that were preferred by a target user during a recommendation session.
- Similarity Arcs connect IEs that refer to the same features, and have weights relative to the similarity between connected IEs. CCRNs makes the assumption that similarities between IEs are always zero, and so these are not included. It should be noted that in Figure 1 only six similarity arcs are shown - a typical CCRN should contain $(n-1)!$ arcs for every n IEs.

When performing collaborative recommendation, the user who is seeking a recommendation is presented to the CCRN as a target node (T in Figure 1). User T has previously liked item 2, which was also liked by User a so those lines have arrows reflecting permanent activation; *in the current recommendation cycle* User T has liked item 4, which was also liked by users a and c, so those relevance arcs are darker and have arrows to reflect temporary activation. As the recommendation session continues, further relevance arcs will be created to express new preferences. Activation is spread across the net structure (shown by the arrowed relevance arcs) and accumulated at the neighbouring nodes. In the example diagram, neighbour a would be returned as the target user's NN with an activation of 2 (c would be in second place with an activation of 1).

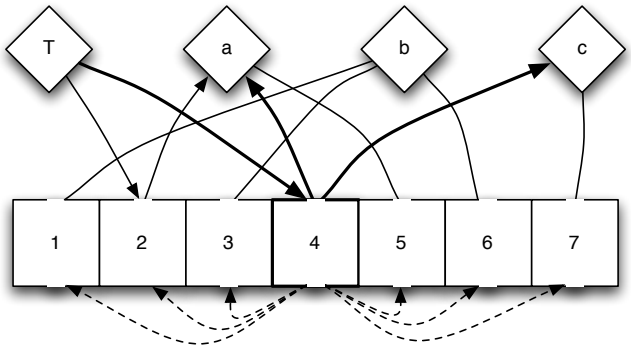


Fig. 1. CCRN Architecture. The diamonds at the top are nodes representing users, where T is the target user. Squares 1 to 7 are IEs representing 7 items that are available for recommendation. The lines connecting IEs to Nodes are relevance arcs, relating items to users that liked them in the past; the broken arrows below the IEs represent similarity arcs from IE 4 to all other IEs in the net, which only exist in traditional CRNs and are shown here only for comparative illustration.

This architecture provides behaviour that is no different from typical collaborative recommendation approaches. However, because of the nature of the CRN structure, it is possible to inject additional information into the retrieval process. We inject information about useful users for recommendations into the "Sticky Layer" of the CCRN.

3.1 The Sticky Layer

Our hypothesis is that in order to maintain a consistent recommendation process towards a stated goal, it is best to reuse recommendations from users that have proven to be responsible for good recommendations during a session so far. The analogy would be that the user is taking advice from a subset of the users that are providing recommendations and following these trusted users' advice over a number of cycles. Thus, the CCRN incorporates a measure of context at each cycle in the recommendation process.

We use a feedback mechanism called the *Sticky Layer*, which makes good neighbours *stick*, so they are more likely to be re-chosen as recommendation partners for the target user during a given recommendation session. This layer adds a memory to the recommendation session and performs a priming function with respect to neighbour selection. The job of the Sticky Layer is to promote or demote neighbours based on their performance as recommendation partners to date in a session. At the end of each cycle, every NN to the target user in that cycle, receives a *sticky boost*, (positive or negative), depending on the quality of the neighbour's *contribution* to the recomendations in the cycle. Sticky boosts are currently in units of 1. The sticky boost can be updated in three ways:

– if the target user selects one of the recommended items as preferable in a cycle and a NN has *contributed* to that item being recommended (that item is in her profile) we increment her sticky boost (by 1).
– Similarly, we can decrement the sticky boost of a NN if she did not contribute to that item being recommended.
– We can also decrement the sticky boost of a NN if she contributed to a set of recommendations made in a cycle in which all of the (bad) recommendations were rejected by the target user.

This boost is incorporated at the Sticky Layer of the CCRN, where it is added to the similarity returned from the CCRN. In this way, neighbours are retrieved from the CCRN using a combination of their weight from the CCRN and their sticky boost. By using both negative and positive contributions to control the boost levels of neighbours we are assuring that only neighbours that are *consistently* performing well will be promoted. Of course the sticky boost levels (as well as the short-term profile, or temporary activations in the CCRN), only stay active during a recommendation session.

4 Evaluation

Our preliminary evaluation aims to eexamine the methods described in this paper, and in particular how (if) the Sticky Layer is helping neighbour selection. The experiments are based on the 100,000 MovieLens Dataset [16], which consists of 100,000 ratings for 1682 movies by 943 users. We select 100 of these users at random as our target users. Each profile consists of a list of movies that the user has seen and a corresponding set of ratings on a scale of 1 - 5, (1 meaning the user did not like the movie and 5 meaning they liked it a lot). This is the same dataset used by Bridge and Kelly [8]. Currently we are using a measure of profile overlap as our profile similarity metric. [1] Since the overlap measure does not require user ratings, we have simplified the MovieLens data by removing all (disliked) profile items with a rating < 3. Therefore each profile is simply a list of previously liked movies. The average profile size is 88 items. This is also in keeping with our previous research that found that negative preferences are not necessarily as useful as positive ones [5]. Of course, the disliked movies would be retained in a real world system to ensure that they would not be recommended.

4.1 Methodology

In this evaluation we make use of simulated artificial users as a real user trial has not been possible yet. The MovieLens dataset contains genre information (lists

[1] This is in contrast to our previous research and indeed much of collaborative recommendation research where the Pearson Correlation Coefficient is used. However we have found that we actually achieve better results using just overlap. This may mean that the ratings in the MovieLens dataset are not reliable, or the reason may lie elsewhere, but certainly this issue deserves further investigation.

of categories, e.g. "comedy, romance"), for the movies, which we use to simulate user feedback in our evaluation, (though of course genre information is *not* used to generate recommendations). A (simulated) user will select the item with the highest overlap of genre categories with the target item as the best recommended item. The user can select an item so long as it has a genre category overlap > 0. Otherwise, if all recommended items in a cycle have a genre category overlap of 0, the user will reject them all. Essentially we are modelling a user's current mood based on genres. Of course there could be many other factors that influence a user's current mood and preferences, genre information is used here because it was a viable way to do the simulation. A *leave-one-out* test is employed to evaluate the search for specific target items in each evaluation trial. In each trial every item in the user profile is in turn used as the target item (during which time it is removed from the profile). In each cycle 3 recommendations are made, from the profiles of 50 NNs. We evaluate the CCRN and the Sticky Layer methods we propose, using a number of different trials:

CCRN Only ($CCRN$) In this trial we use the CCRN by itself with no Sticky Layer. This performs equally to the standard conversational collaborative recommendation which is to be expected, and serves as our benchmark here.

CCRN with Sticky Layer ($CCRN + S$) Here we examine using the Sticky Layer in conjunction with the CCRN, and consider different combinations of rewards and penalties for neighbour contributions in a recommendation session. We look at four variations, the first where neighbours are only rewarded ($CCRN + S_{rewards}$), the second where the neighbours are both rewarded, and penalised if they contribute to a set of bad items being recommended ($CCRN + S_{rewardsAndPenaliseBadItems}$), the third when the neighbours are both rewarded, and penalised if they don't contribute to a liked item being recommended ($CCRN + S_{rewardsAndPenaliseGoodItem}$), and finally the fourth where all three are combined ($CCRN + S_{rewardsAndAllPenalties}$). Note that when we refer to $CCRN + S$ we are referring to all four techniques in general. Refer to Section 3.1 for more details.

Bootstrapped Sticky Layer(S) Here we use the Sticky Layer by itself without the CCRN, except at the very start of a session when we use the CCRN until 5 pieces of information have been added to the Sticky Layer, in order to avoid it having to choose neighbours completely at random. So although the Sticky Layer in these experiments receives some help at the start, we would argue that this bootstrapping is minimal. Note that we would not expect this technique to perform better than the ones always using the CCRN since it cannot use any of the main long-term profile information contained in the CCRN. We test it by itself here to gain a more exact idea of how it performs. As with the combined CCRN and Sticky Layer approach we again test four variations of this technique ($S_{rewards}$, $S_{rewardsAndPenaliseBadItems}$, $S_{rewardsAndPenaliseGoodItem}$, and $S_{rewardsAndAllPenalties}$), (and S refers to all four techniques in general).

CCRN with Persistent Sticky Layer ($CCRN + S_{Persist}$) This uses a Sticky Layer that is persistent across sessions. Instead of clearing the sticky

information after a given session it is retained for future sessions. This is testing how strong our theory is that a user's current preferences are controlled by her current mood and that certain information is only relevant for a limited amount of time.

Random ($SS - CR_{Random}$) Here we run a trial where the NNs are selected at random and the results are averaged over 10 runs (SS-CR = C-CR if neighbours are selected at random). We use this as a final lower benchmark.

In each version the system is evaluated according to three different search criteria. Each one measures the average session length, which equates to recommendation quality.

Item the actual target item needs to be found (recommended) in order to count as a success, i.e. in order for a session to be completed satisfactorily.

Genre an item with the same genre categories as the target item needs to be found.

SimGenre an item with similar genre categories to that of the target needs to be found. (We define two sets of genre categories to be similar if they have an overlap value of $>= 0.5$).

4.2 Results

The results for each of our different system variations, against each of our different success criteria, are presented in Table 1. They report the average length of session needed before the success criterion is found (smaller values are better). The results show that there is little or no difference between the trials that use a Sticky Layer with the CCRN ($CCRN + S$), and the baseline trial that only uses the CCRN, ($CCRN$). This is disappointing as we would have expected that the Sticky Layer would improve the quality of the neighbours selected and ultimately reduce the session lengths. However, this may be due to the Sticky Layer not being given enough influential weight.

When we compare the trial where the Sticky Layer is persistent ($CCRN + S_{Persist}$) to when it is not, we find that it performs significantly worse. This is to be expected and further supports our argument that a user's *current* preferences can be different from her more long-term static preferences [5].

If we compare how the Boostrapped Sticky Layer (S), performs (with little influence from the CCRN) two things are clear. Firstly, penalising users because they did not contribute to a liked item being recommended is too harsh a strategy to be effective. This is evident from looking at the techniques that penalise neighbours for not contributing to a liked item being recommended ($S_{rewardsAndPenaliseGoodItem}$ and $S_{rewardsAndAllPenalties}$) which perform no better than the random technique. Conversely, if we don't penalise users in this manner in the Sticky Layer ($S_{rewards}$ and $S_{rewardsAndPenaliseBadItems}$), we can achieve results that far outperform the random technique, and that are similar to $CCRN + S$ techniques. So although the addition of the Sticky Layer is not improving the CCRN results, there is certainly some value in using it. Moreover,

Table 1. Experimental Results

	Item	**Genre**	**SimGenre**
$CCRN$	162.2	53.92	11.98
$CCRN + S_{rewards}$	161.91	53.95	12.01
$CCRN + S_{rewardsAndPenaliseBadItems}$	160.62	53.46	11.93
$CCRN + S_{rewardsAndPenaliseGoodItem}$	163.22	53.89	11.98
$CCRN + S_{rewardsAndAllPenalties}$	162.05	53.43	11.89
$S_{rewards}$	162.98	53.69	11.97
$S_{rewardsAndPenaliseBadItems}$	164.47	54.36	12.7
$S_{rewardsAndPenaliseGoodItem}$	220.94	65.37	12.21
$S_{rewardsAndAllPenalties}$	220.28	64.97	12.78
$CCRN + S_{Persist}$	180.52	60.19	13.03
$SS - CR_{Random}$	223.85	68.68	14.94

the very fact that the Sticky Layer can achieve results comparable the CCRN may be promising in dealing with the *cold-start* problem faced by collaborative recommender systems [17]. The problem arises when a new user starts to use the system and there is no preference information for her yet, making recommendation difficult. Similarily, our temporary CCRN activations (short-term profile) and Sticky Layer are both empty too at the start of a recommendation session.

5 Conclusions

We propose that *sticking* with users that are contributing positively to the recommendation process is a novel and useful way of increasing recommendation accuracy and reducing session length. Our preliminary evaluations have shown that Sticky Layer recommendation performs better than random recommendation and confirms that our approach has merit. While our results are not overwhelmingly successful, we propose that it may have some value in the cold-start problem of initial recommendations (even if it requires some minimal bootstrapping, it could help speed up the delay while a user is building up enough preference history).

The second contribution of this work is the implementation of a novel data structure called the Collaborative Case Retrieval Net (CCRN) for retrieving NNs for collaborative recommenders. This work follows on from earlier work in the CBR and ACF fields by Hayes *et al.* [9] and Delany *et al.* [13]. CCRNs offer an efficient and fast way of retrieving useful neighbours and allow the incorporation of additional context data into the recommendation process. We demonstrate this with the sandwiching of the Sticky Layer onto the CCRN.

In the future we will investigate the individual usefulness of the Sticky Layer and the temporary and permanent components of the CCRN, by adjusting their relative weights. We will also investigate the use of a decay function to prevent certain users from leading the recommendation process after they have outlived their usefulness. Finally a live user trial is in the pipeline.

References

1. Rafter, R., Bradley, K., Smyth, B.: Automated Collaborative Filtering Applications for Online Recruitment Services. In: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Trento, Italy, Springer-Verlag (2000) 363 – 368
2. Bridge, D.: Towards Conversational Recommender Systems: A Dialogue Grammar Approach. In: Proceedings of the Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Sixth European Conference in Case-Based Reasoning, Aberdeen, Scotland (2002) 9–22
3. Goker, M., Thompson, C.: The Adaptive Place Advisor: A Conversational Recommendation System. In: Proceedings of the 8th German Workshop on Case Based Reasoning, Lammerbuckel, Germany (2000)
4. Aha, D.W., Breslow, L.A., Muñoz-Avila, H.: Conversational Case-based Reasoning. Applied Intelligence **14**(1) (2001) 9–32
5. Rafter, R., Smyth, B.: Conversational Collaborative Recommendation - An Experimental Analysis. Artificial Intelligence Review **24**(3–4) (2005) 301 – 308
6. McGinty, L., Smyth, B.: Evaluating Preference-Based Feedback in Recommendation Systems. In: Proceedings of the 13th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2002), Limerick, Ireland, Springer-Verlag (2002) 209–214
7. Hayes, C., Cunningham, P.: Context Boosting Collaborative Recommendations. Knowledge-Based Systems **17**(2–4) (2004) 131–138
8. Bridge, D., Kelly, J.P.: Ways of Computing Diverse Collaborative Recommendations. In Wade, V., Ahsman, H., Smyth, B., eds.: Adaptive Hypermedia and Adaptive Web-Based Systems, Springer (2006) 41–50
9. Hayes, C., Cunningham, P., Smyth, B.: A Case-Based Reasoning View of Automated Collaborative Filtering. In: ICCBR '01: Proceedings of the 4th International Conference on Case-Based Reasoning, Vancouver, British Columbia, Canada, Springer-Verlag (2001) 234–248
10. Billsus, D., Pazzani, M.: A Hybrid User Model for News Story Classification. In: Proceedings of 7th International Conference On User Modelling (UM99), Banff, Canada (1999) 99–108
11. Widyantoro, D.H., Ioerger, T.R., Yen, J.: An Adaptive Algorithm for Learning Changes in User Interests. In: Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM '99), Kansas City, Missouri, ACM Press (1999) 405–412
12. Lenz, M., Auriol, E., Manago, M. In: Diagnosis and Decision Support. Springer (1998) 51–90
13. Delany, S., Cunningham, P., Coyle, L.: An Assessment of Case-Based Reasoning for Spam Filtering. Artificial Intelligence Review **24**(3–4) (2005) 359–378
14. Richter, M.M.: Introduction. In Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S., eds.: Case-Based Reasoning Technology, From Foundations to Applications, Springer (1998) 1–16
15. Doyle, D., Loughrey, J., Nugent, C., Coyle, L., Cunningham, P.: FIONN: A Framework for Developing CBR Systems. Expert Update **8**(1) (2004) 11–14
16. GroupLens: MovieLens Dataset. (http://www.grouplens.org/)
17. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating Word of Mouth. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Denver, Colorado, ACM Press (1995) 210–217

Author Index